Trellis Decoding Complexity of Linear Block Codes*

A. B. Kiely S. Dolinar R. J. McEliece L. Ekroot W. Lin

December 15, 1994

Abstract

We consider the problem of finding a trellis for a linear block code that minimizes one or more measures of trellis complexity. The domain of optimization may be different permutations of the same code, or different codes with the same parameters. Constraints on trellises, including relationships between the minimal trellis of a code and that of the dual code, are used to derive bounds on complexity. We define a partial ordering on trellises: if a trellis is optimum with respect to this partial ordering, it has the desirable property that it simultaneously minimizes all of the complexity measures examined. We examine properties of such optimal trellises and give examples of optimal permutations of codes, most notably the (48,24,12) quadratic residue code.

Keywords– trellis decoding, block codes, decoding complexity

*The research described in this paper was carried out in the Communications Systems Research Section of the Jet Propulsion Laboratory, California institute of Technology, under a contract with the National Aeronautics and Space Administration. A portion of W. Lin's contribution was sponsored by AFOSR grant no. F4960-94-1-005. Part of this work was presented at the 32nd Annual Allerton Conference on Communication, Control, and Computing, Allerton, Illinois, October 1994.

A. B. Kiely, S. Dolinar, and L. Ekroot are with the Jet Propulsion Laboratory, California institute of Technology, Pasadena, CA 91109.

R. J. McEliece is with the Jet Propulsion Laboratory and the California Institute of Technology, Pasadena, CA 91125

W. Lin is with the California Institute of Technology, Pasadena, CA 91125.

1 Introduction

Every linear block code can be represented by a *minimal trellis*, originally introduced by Bahl *et. al.* [1], which is a labeled graph that can be used as a template for encoding or decoding. As shown by McEliece [26, 27], the minimal trellis simultaneously minimizes the maximum number of states, the total numbers of vertices and edges in the trellis, and the total numbers of additions and path comparisons required for decoding with a Viterbi algorithm.

A code's minimal trellis is unique as long as the ordering of the code's symbols is fixed. However, different permutations of the symbols yield different minimal trellises. An optimum minimal trellis for the code is one which minimizes a suitable measure of trellis complexity over all possible permutations of the code. There are no known efficient algorithms for constructing optimum minimal trellises.

Using [27] as a starting point, we examine properties of the minimal trellis representation of a code and its dual for a fixed permutation, and use these results to examine the problem of finding a permutation that minimizes one or more trellis complexity measures. We extend these results to the problem of finding a minimal complexity trellis over all codes with the same parameters. We identify certain sufficient conditions for a code or a permutation to simultaneously minimize all of the complexity measures.

Section 2 reviews the subject of minimal trellises for a fixed permutation of a code. We examine the building blocks of such trellises, and identify several different measures of trellis size or complexity.

In Section 3 we illustrate the connection between the minimal trellis of a code and that of the dual code which provides the foundation for duality relationships that appear throughout this paper. The section includes results that describe the structure and complexity of trellises for self-dual and other special codes.

In Section 4 we discuss dimension/length profiles of a code [14, 10, 30], which are equivalent to Wei's generalized hamming weights [32]. The dimension/length profiles are used to derive some straightforward complexity bounds, We summarize some properties of these profiles including duality relationships.

We define a partial ordering on minimal trellises in Section 5. If the minimal trellises for two codes are comparable in terms of this partial ordering, then each of the complexity measures for one trellis is bounded by the same measure evaluated for the other trellis. This partial ordering can sometimes be used to identify the permutation of a code with the least (or most) complex minimal trellis, or the code with the lowest (or highest) complexity trellis of all codes with the same parameters. The extremal codes determined by this partial ordering turn out to meet the complexity bounds described in Section 4. We illustrate certain properties and give examples of such permutations and codes.

Finally, wc give some concluding remarks in Section 6.

2 Minimal Trellis Representation of a Code

2.1 The Minimal Span Generator Matrix

For any linear (n,k) block code $\mathcal{C}$ over $GF(q)$ there exists a minimal span generator matrix (MSGM) representing $\mathcal{C}$. A minima] trellis 7 for the code can be constructed from the MSGM. The trellis has $n+1$ levels of vertices and n levels of edges. The vertex levels, called *depths*, arc numbered from 0 to n; the edge levels, called *stages*, are numbered from 1 to n. Each stage of edges corresponds to onc stage of encoding or decoding using the trellis. Each vertex at depth i represents a possible encoder st ate after the iii' stage of encoding. The i^{th} stage corresponds to the i^{th} column of the generator matrix, whereas the i^{th} depth corresponds to the "space between" columns i and $i+1$.

The *edge-span* of any row of the generator matrix is the smallest set of consecutive integers (stages) containing its nonzero positions. The *vertex-span* of the row is the set of depths i such that at least one nonzero symbol occurs before and after depth i. Using the generator matrix to encode k information symbols in n stages of encoding, the edge-span of the j^{th} row represents the interval of stages during which tbc j^{th} information symbol can affect the encoder **output.** The **vertex-span** of the j^{th} row is the set, of depths at which the j^{th} information symbol can affect the encoder state.

For example, tbc (6,3,3) shortened Hamming code has minimal span generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{1}$$

The edge spans arc {1,2,3}, {2,3,4,5,6}, and {3,4,5}. The vertex spans arc {1,2}, {2,3,4,5}, and {3,4}. Wc usc the term *span length* to refer to the cardinality of a span.

A remarkable result is that the MSGM simultaneously makes all of the spans as short as possible: the edge-sl)ans (vertex-spans) for any other generator matrix representing $\mathcal{C}$ always contain the corresponding spans of some row-permuted MSGM [27]. Any generator mat rix can be put into minimal span form using the following greedy algorithm: at each step, perform any row operation that reduces the edge-span of any row of the matrix. The rows of the MSGM arc then "atomic codeword s," according to the terminology of Kschischang and Sorokine [21].

Each vertex or state at a given depth can be uniquely labeled using k or fewer symbols from $GF(q)$. But any given state-label symbol can be reused to represent several information symbols, as long as the vertex-spans of the corresponding rows of the generator matrix do not, overlap. This reassignment of state-label symbols to multiple rows of the generator matrix is the key to efficient trellis representations of the code.

Example 1 The minimal trellis T produced for the (6.3,3) shortened Hamming with MSGM

given in equation (1) is shown in Figure 1. For this trellis we can define the binary state label to be $s_2 s_1$, where $s_2 = 1$ at depth i if the second information bit is 1 and r' is within tbc vertex-sl)an of the second row; and $s_1 = 1$ if either (a) the first information bit is 1 and i is within tbc vertex-span of the first row, or (b) the third information bit is 1 and i is within the vertex-sJJan of the third row. This time-sharing arrangement for state bit s_1 is possible because the vertex spans of the first and third rows do not overlap. ❑

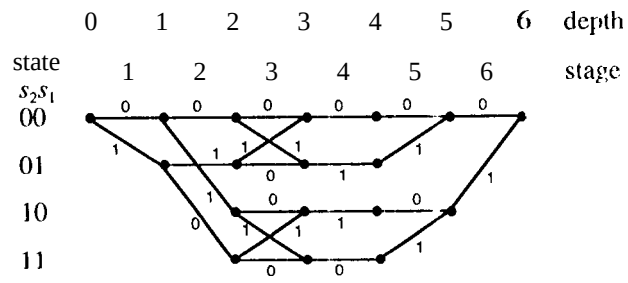


Figure 1: A minimal trellis for the (6,3,3) shortened Hamming code.

In the sequel we will be primarily interested in *nondegenerate codes,* which *we* define as codes whose minimum distance d and dual code minimum distance $d^\perp$ arc both at least 2. Degenerate codes have a simple interpretation: If $d < 2$, the vertex-sl)an of some row of the generator matrix must be empty; if $d^\perp < 2$, some column of the generator matrix must be identically zero. For a degenerate code, we can simply ignore the extraneous symbol positions (if $d^\perp < 2$) and/or separately decode the unprotected information symbols (if $d < 2$). The code consisting of the remaining code symbols is then nondegenerate.

2.2 Past and Future Subcodes

Following Forney [8], let us define the i^{th} past, and future subcodes, denoted $\mathcal{P}_i$ and $\mathcal{F}_i$, to be tbc sets of all codewords whose vertex-spans arc contained in [O, i — 1] and [i + 1, n] respectively. These subcodes arc nested in the following manner:

$$\{0^n\} = \mathcal{P}_1 \subseteq \mathcal{P}_2 \subseteq \cdots \subseteq \mathcal{P}_n = \mathcal{C}$$

$$\mathcal{C} = \mathcal{F}_1 \supseteq \mathcal{F}_2 \supseteq \cdots \supseteq \mathcal{F}_n = \{0^n\}.$$

Each of these subcodes is linear: $\mathcal{P}_i$ is generated by the rows of the MSGM whose vertex-spans arc contained in [0, i — 1], and $\mathcal{F}_i$ is generated by the rows of tbc MSGM whose vertex-spans arc contained in [i + 1, n]. Consequently, the dimensions of these codes can be easily determined from the MSGM: $f_i \triangleq \dim(\mathcal{F}_i)$ is the number of rows for which t he leftmost nonzero entry lies in column $i + 1$ or later, and $p_i \triangleq \dim(\mathcal{P}_i)$ is the number of rows for which the rightmost nonzero entry lies in

column r' or earlier [27]. This implies that p_i and f_i are monotonic,

$$0 = p_0 \leq p_1 \leq \ldots \leq p_n = k \tag{2}$$

$$k = f_0 \geq f_1 \geq \ldots \geq f_n = 0 \tag{3}$$

and never change by more than 1 from one index to the next

$$p_i \leq p_{i-1} + 1, i = 1, \ldots, n \tag{4}$$

$$f_{i-1} \leq f_i + 1, \ i = 1, \ldots, \ n. \tag{5}$$

For each $1 \leq i \leq$ n, we define the *left-* and *right-basis indicators*, $l_i, r_i \in \{0, 1\}$, to identify the positions where the future and past dimensions change:

$$l_i \triangleq f_{i-1} - f_i$$

$$r_i \triangleq p_i - p_{i-1}.$$

For any i, $l_i = 1$ if and only if the edge-span of some row of the MSGM G *begins in* column i, or equivalently, the i^{th} column of G is linearly independent of the i – 1 columns to the *left*. Similarly $r_i = 1$ if and only if the edge-span of some row of G *ends* in column i, i.e., the i^{th} column of G is linearly independent of the $n - i$ columns to the *right*. The columns where $l_i = 1$ and the columns where $r_i = 1$ each form a basis for the column space of G, and these sets are called the *left basis* and the *right basis*, respectively. The positions of the left and right basis columns can be regarded as information positions when the generator matrix is used to encode the information left-to-right or right-to-left, respectively.

2.3 Primitive Structures of a Minimal Trellis

There are four basic building blocks that can be used to construct the minimal trellis for any nondegenerate code. At any given stage i, all primitive structures are of the same type, which is determined by the values of l_i and r_i. The primitive structures are:

1. Simple extension (–): This primitive structure appears at stage i when $l_i = 0$, $r_i = 0$, e.g., stage 4 in Figure 1. Simple extensions at stage i imply a single edge out of each vertex at depth r' — 1 and a single edge into each vertex at depth i, hence the number of vertices remains constant.

2. Simple expansion (<): corresponds to $l_i = 1$, $r_i = 0$, e.g., stages 1 and 2 in Figure 1. '1' here arc q *edges* out of each out of each vertex at depth i - 1, and a single edge into each vertex at depth i, hence multiplying by q the number of states from one vertex depth to the next.

3. Simple merger (>): corresponds to $l_i = 0, r_i = 1$, e.g., stages 5 and 6 in Figure 1. A simple merger is a time-reversed simple expansion, reducing the number of states by a factor of g.

4. Butterfly (x): corresponds to $l_i = 1$, $r_i = 1$, e.g., stage 3 in Figure 1. '1'here arc g edges out of each vertex at depth $i - 1$ aud q edges iuto each vertex at depth i, hence the number of states is constant.

The total numbers of such primitive structures in the trellis are denoted by N. , $N_<$, N>, aud N_x, respectively. For example, the trellis in Figure 1 has $N_< = 3 = N>$, $N_x = 2, N_- = 4$. Because the graph has exactly one initial node and one terminal node, the total number of simple expansions must equal the total number of simple mergers:

$$N_< = N>.$$

The total number of edges in the trellis, E, can be found by counting the number of edges associated with each primitive trellis structure:

$$E = N- + qN_< + qN_> + q^2 N_x . \tag{6}$$

Similarly, the total number of mergers M is the sum of the number of simple mergers aud the mergers included in butterflies

$$M = N> + qN_x \tag{7}$$

If we count the total number of vertices associated with each primitive. structure, then each vertex in the trellis (excluding initial and terminal nodes) will be counted twice, so the total number of vertices V satisfies

$$2V - 2 = 2N_- + (q + 1)N_< + (q\text{-}1)N_> + 2qN_x$$

which gives

$$V = 1 + N_- + (q + 1)N_< + qN_x . \tag{8}$$

Combining *(6), (7),* and (8) we find

$$M = \frac{E - V + 1}{q - 1}.$$

This is the generalization of the binary version of this result fouud in [27].

2.4 Measures of Trellis Complexity for Viterbi Decod ng

The vertex space dimension at depth i is

$$v_i = k - f_i - p_i, \quad i = 0, \ldots, n \tag{9}$$

aud the edge spat.c dimension at stage i is

$$c_i = k - f_i - p_{i-1}, \quad i = 1, \ldots, n. \tag{10}$$

The total number of vertices at depth i is q^{v_i} and the total number of edges at stage i is q^{e_i}. Of course $v_i \geq 0$ for all i since at least one vertex must exist at each depth. Also, for nondegenerate codes, $e_i \geq 1$ for all i, i.e., no stage consists of a single edge.

The most commonly used measure of Viterbi decoding complexity for a minimal trellis is the maximum dimension of its state space,

$$s_{\max} \triangleq \max_i v_i. \tag{11}$$

This complexity metric has been cited as one of the essential characteristics of any code [28]. Similarly, the maximum dimension of the edge space is

$$e_{\max} \triangleq \max_i e_i. \tag{12}$$

Forney argues that this is a more relevant complexity measure because, unlike $s_{\max}$, this quantity cannot be reduced by combining adjacent stages of a trellis [10].

A different metric, used in the derivation of the MSGM [27], is the total length of all the edge-spans of the rows of the MSGM:

$$\varepsilon \triangleq \sum_{j=1}^{k} \varepsilon_j \tag{13}$$

where ε_j denotes the length of the edge-span of the j^{th} row of the MSGM. A similar span length metric is the total length of all the vertex-spans:

$$\nu \triangleq \sum_{j=1}^{k} \nu_j$$

where $\nu_j = \varepsilon_j - 1$ is the length of the vertex-span of the j^{th} row of the MSGM. These two metrics are equivalent to the sums of all the edge dimensions or vertex dimensions (summed over stages or depths, respectively):

$$\nu + k = \varepsilon = \sum_{i=1}^{n} e_i = k + \sum_{i=0}^{n} v_i.$$

It is argued in [26, 27] that more meaningful measures of Viterbi decoding complexity are the total number of edges E, vertices V, and mergers M, rather than simply the vertex or edge dimensionality:

$$E = \sum_{i=1}^{n} q^{e_i} \tag{14}$$

$$V = \sum_{i=0}^{n} q^{v_i} \tag{15}$$

$$M = \sum_{i=1}^{n} r_i q^{v_i} = \sum_{i=1}^{n} l_i q^{v_{i-1}} = \frac{1}{q} \sum_{i=1}^{n} l_i q^{e_i} = \frac{1}{q} \sum_{i=1}^{n} r_i q^{e_i}. \tag{16}$$

E is equal to the number of binary additions required to compute path metrics, and M is the number of q-ary comparisons required to merge trellis paths. The computational complexity of Viterbi decoding is proportional to E [27].

3 Minimal Trellis Representation of the Dual Code

In this section we explore the relationship between the minimal trellises for a code $\mathcal{C}$ and its dual $\mathcal{C}^\perp$.

3.1 Past and Future Subcode Relationships

As discussed in Section 2.2, $l_i = 0$ if and only if the i^{th} column of the MSGM can be written as some linear combination of the i – 1 columns to its left. In other words, there exists a dual codeword y of the form

$$y = \underbrace{XXX\cdots X}_{i-1} 1 \underbrace{000\cdots 0}_{n-i}$$

Where $XXX\ldots X$ denotes some sequence of symbols from $GF(q)$. Defining $y_1, y_2, \ldots, y_{n-k}$ in this manner for each of the left-dependent columns in the MSGM produces $n-k$ dual codewords of the form

$$\begin{aligned} y_1 &= XXX\cdots X1000\cdots 0 \\ y_2 &= XXX\cdots X1000\cdots 0 \\ &\vdots \\ y_{n-k} &= XXX\ldots X1 \end{aligned}$$

These **dual codewords are clearly linearly** independent and thus can be used as the rows of the generator matrix for $\mathcal{C}^\perp$. We see that the positions where $r_i^\perp = 1$ are precisely the positions where $l_i = 0$; the same argument applied to the right-dependent columns shows that the positions where $l_i^\perp = 1$ are precisely the positions where $r_i = 0$. Here $l_i^\perp$ and $r_i^\perp$ are the left- and right-basis indicators for $\mathcal{C}^\perp$. These observations lead to the following theorem.

Theorem 1 *For each $0 \le i \le n$, the left- and right-basis indicators for a code and its dual are related by*

$$l_i + r_i^\perp = l_i^\perp + r_i = 1$$

and the dimensions p_i, f_i, of the past and future subcodes of a code are given in terms of those of the dual code $p_i^\perp$, $f_i^\perp$ as follows:

$$p_i = k - n + i + f_i^\perp$$

$$f_i = k - i + p_i^\perp.$$

We believe that this result, which relates minimal trellises of a code and dual for any *fixed* permutation, is more fundamental than similar dual relationships for permutations of codes. This result is also contained in [10], but derived by first considering permutations of codes.

3.2 Primitive Trellis Structures for the Dual Code

Much information about the trellis for the dual code can be inferred from the trellis structure of the code. For example, if the code has a simple expansion at the i^{th} stage, then $l_i = 1, r_i = 0$, which implies, using Theorem 1, that the dual code has $l_i^\perp = 1, r_i^\perp = 0$, hence the trellis of the dual code also has a simple expansion structure at this stage. Repeating this procedure we find the "dual" of each primitive structure, shown in Table 1.

Code Structure	Dual Structure
Simple Extension (–)	Butterfly (⧖)
$l_i = 0, r_i = 0$	$l_i^\perp = 1, r_i^\perp = 1$
Simple Expansion (<)	Simple Expansion (<)
$l_i = 1, r_i = 0$	$l_i^\perp = 1, r_i^\perp = 0$
Simple Merger (>)	Simple Merger (>)
$l_i = 0, r_i = 1$	$l_i^\perp = 0, r_i^\perp = 1$
Butterfly (⧖)	Simple Extension (–)
$l_i = 1, r_i = 1$	$l_i^\perp = 0, r_i^\perp = 0$

Table 1: Dual primitive structures.

Given an unlabeled trellis, Table 1 can be used to determine the number and type of primitive structures present at every depth of the trellis for the dual code. However, as the following example illustrates, we cannot in general determine the interconnections without additional information about the code.

Example 2 Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the codes with generator matrices $G_1 = \begin{bmatrix} 0110 \\ 1111 \end{bmatrix}$ and $G_2 = \begin{bmatrix} 0110 \\ 1101 \end{bmatrix}$ respectively. From Figure 2, which shows the minimal trellises for these codes and their duals, we can see that $\mathcal{T}(\mathcal{C}_1)$ and $\mathcal{T}(\mathcal{C}_2)$ have the same structure (only the edge labels are different), but $\mathcal{T}(\mathcal{C}_1^\perp)$ and $\mathcal{T}(\mathcal{C}_2^\perp)$ do not. u

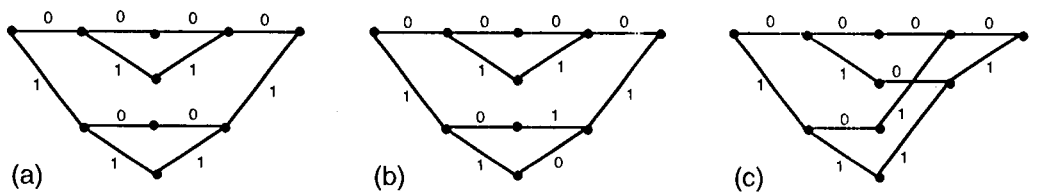


Figure 2: Minimal trellises (a) $\mathcal{T}(\mathcal{C}_1)$ and $\mathcal{T}(\mathcal{C}_1^\perp)$ ($\mathcal{C}_1$ is self-dual), (b) $\mathcal{T}(\mathcal{C}_2)$, (c) $\mathcal{T}(\mathcal{C}_2^\perp)$.

The dual relationship for primitive structures shown in Table 1 implies that

$$N_<^\perp = N_< = N_> = N_>^\perp$$

and

$$N_- = qN_{\mathbf{X}}^{\perp}.$$

3.3 Dual Code Complexity Measures

The following well-kuow]l result, first noted by Forney [8], is a consequence of (9) aud Theorem 1.

Lemma 1 *A code and its dual code have equivalent vertex spaces, namely for each i,*

$$v_i = v_i^{\perp}.$$

Consequently, many of the trellis complexity measures for a code can be determined by evaluating the same measure on the dual code:

$$V = V^{\perp}$$

$$s_{\max} = s_{\max}^{\perp}$$

$$\varepsilon - k = \nu = \nu^{\perp} = \varepsilon^{\perp} - (n - k).$$

Note that this implies $\varepsilon = \varepsilon^{\perp}$ for auy rate $\frac{1}{2}$ code.

The number of edges in the minimal trellis of a code aud its dual arc not as conveniently related. From (1 O) aud Theorem 1,

$$e_i = e_i^{\perp} + (1 - r_i^{\perp} - l_i^{\perp})$$

for each $1 \leq \mathrm{i} \leq n$. Consequently, since $|\ 1 - r_i^{\perp} - l_i^{\perp}| \leq 1$, and from the definition of E,

$$\frac{1}{q}E \leq E^{\perp} \leq qE.$$

Equality is possible only for the degenerate (n, n, 1) code or its dual

3.4 Minimal Trellises for Self-Dual and Other Special Codes

For self-dual codes, the theory of the previous two sections collapses neatly to yield stronger results because for any such code ii = $l_i^{\perp}$ aud $r_i = r_i^{\perp}$ for all i. Consequently from Theorem 1:

Theorem 2 For *any self-dual code* C, *for each* $i = 1,2, \ldots n$, *either:*

1. $l_i = 1$ *and* $r_i = O$, *or*

2. $l_i = 0$ *and* $r_i = 1$,

i. c., every stage corresponds to an information symbol when encoding from one direction and a parity symbol when encoding from the other direction. The only primitive trellis structures in $\mathcal{T}(C)$ *arc simple expansions and simple mergers.*

We say that two trellises $\mathcal{T}_1, \mathcal{T}_2$ have equivalent structures, denoted $\mathcal{T}_1 \sim \mathcal{T}_2$ if $\mathcal{T}_1$ can be made identical to $\mathcal{T}_2$ by an appropriate relabeling. E.g., for the codes of example 2, $\mathcal{T}(\mathcal{C}_1) \sim \mathcal{T}(\mathcal{C}_2)$ but $\mathcal{T}(\mathcal{C}_1^\perp) \not\sim \mathcal{T}(\mathcal{C}_2^\perp)$.

The converse of Theorem 2 does not hold: a code whose minimal trellis contains only simple expansions and mergers need not be self-dual. In fact, C may not be self-dual even when $\mathcal{T}(C) \sim \mathcal{T}(\mathcal{C}^\perp)$, e.g., the code whose MSGM is $G = \begin{bmatrix} 1010 \\ 0111 \end{bmatrix}$. However, a relabeled trellis for this code is self-dual.

Lemma 2 *Given any binary code $\mathcal{C}$ such that $\mathcal{T}(\mathcal{C})$ contains only simple expansions and simple mergers, there exists a self-dual code $\mathcal{C}'$ such that $\mathcal{T}(\mathcal{C}) \sim \mathcal{T}(\mathcal{C}')$.*

Proof: Let G denote an MSGM for $\mathcal{C}$. Form a $k \times n$ MSGM G' for $\mathcal{C}'$ by setting each element of G' to 1 if the corresponding element in G begins or ends a span, or zero otherwise. Then $\mathcal{T}(C) \sim \mathcal{T}(C')$ and C' is a self-dual code with minimum distance two. ■

The following theorem, which is a consequence of Theorem 2 and equations (6), (7), and (8), shows that for self-dual codes, the complexity measures E, V, and M are linearly related, and the maximum edge and vertex dimensions are equal.

Theorem 3 *For any self-dual code,*

$$V = \frac{q+1}{2q} E + 1$$

$$M = \frac{1}{2q} E$$

$$s_{\max} = c_{\max}.$$

There is another case where we can restrict the type of structures that can appear in the trellis for a code:

Theorem 4 *If $\mathcal{C}$ is such that all codeword weights are divisible by some integer $m > 2$, then*

1. *There does not exist a position i such that $l_i = r_i = 1$, i.e., $\mathcal{T}(\mathcal{C})$ contains no butterfly structures.*
2. *$\mathcal{C}$ cannot have rate greater than $\frac{1}{2}$.*
3. $c_{\max} = s_{\max}$
4. $V \geq \left(\frac{q+1}{q}\right) E + 1$
5. $M \leq \frac{1}{2q} E$

6. $V^{\perp} \leq \left(\frac{q+1}{q}\right) E^{\perp} + 1$

7. $M^{\perp} \geq \frac{1}{2q} E^{\perp}$

Proof: If $l_i = r_i = 1$ then the i^{th} column begins and ends spans in the MSGM. This implies the existence of codewords of the form $x = XXX \ldots X10^{n-i}$ and $y = 0^{i-1}(-1)XXX \cdots X$, where (-1) denotes the additive inverse of 1 in $GF(q)$ and $XXX \cdot$. X denotes some string of symbols in $GF(q)$. Then $x + y$ is a codeword of weight $|x| + |y| - 2$ which cannot be divisible by j. This proves 1. From 1 we have $l_i + r_i \leq 1$ for all i, so $2k = \sum_{i=1}^{n}(l_i + r_i) \leq \sum_{i=1}^{n} 1 = n$, which Proves 2. The fact that $T(\mathcal{C})$ can have no butterfly structures proves 3. From (16), $2qM = \sum_{i=1}^{n}(l_i + r_i)q^{e_i} \leq \sum_{i=1}^{n} q^{e_i} = E$, proving 5, and 4 follows directly. Since $l_i + r_i \leq 1$, Theorem 1 implies $l_i^{\perp} + r_i^{\perp} \geq 1$, which gives 6 and 7. ■

Codes for which all codeword weights are divisible by some integer other than one are called *divisible codes* [31]. Examples of divisible codes include the (31,10,12) cyclic codes and doubly-even self dual codes such as the extended Golay code.

The converse of Theorem 4 does not hold- a code is not necessarily divisible when $l_i + r_i \leq 1$ for all i, e.g., code $\mathcal{C}_1$ of Example 2. If a code and its dual satisfy the conditions of Theorem 4 then the code strongly resembles a self-dual code: the code must have rate $\frac{1}{2}$, and its trellis contains only simple expansions and simple mergers.

4 Trellis Complexity Bounds

Although the results of the previous sections assume a fixed coordinate ordering for the code, the trellis structure, and hence trellis complexity, depends on the permutation of the code. Massey refers to the procedure of re-ordering the code symbols to reduce the trellis complexity as "the *art* of trellis decoding" [25, p. 9].

In this section we identify code parameters that affect the possible trellis complexity, describe upper and lower bounds based on these parameters, and illustrate properties of certain codes that have low complexity trellises.

First, some notation. Let $\mathcal{S}_n$ denote the set of all permutations of $\{1,2, \ldots n\}$, and for any $\pi \in S_n$, let $\mathcal{C}\pi$ denote the code $\mathcal{C}$ with coordinates re-ordered according to π. Because the code and dual code provide symmetric constraints on the code's minimal trellis, the complexity bounds are developed by considering the characteristics of both the code and its dual. We refer to an (n, k, d) code with dual distance $d^{\perp}$ as an $(n, k, d, d^{\perp})$ code.

4.1 Bounds Relating One Complexity Measure to Another

The following lemma arises from the definition of $s_{\max}$ aud $e_{\max}$, and the fact that the vertex and edge dimensions change by 110 more than one unit from one index to the next.

Lemma 3 *The vertex dimensions and edge dimensions are upper bounded by*

$$v_i \le \min\{i, n-i, s_{\max}\}, \quad 0 \le i \le n$$

$$e_i \le \min\{i, n+1-i, e_{\max}\}, \quad 1 \le i \le n$$

Summing the inequalities in Lemma 3 leads to the following bounding relationships among the complexity measures.

Theorem 5 *The total complexity measures ν, ε, V, E are upper boun ded in terms of the maximum complexity measures $s_{\max}, e_{\max}$ by*

$$\nu \le s_{\max}(n - s_{\max}) \tag{17}$$

$$\varepsilon \le e_{\max}(n + 1 - e_{\max}) \tag{18}$$

$$V \le \left[n + \frac{q+1}{q-1} - 2s_{\max} \right] q^{s_{\max}} - \frac{2}{q-1} \tag{19}$$

$$E \le \left[n + \frac{2q}{q-1} - 2e_{\max} \right] q^{e_{\max}} - \frac{2q}{q-1} \tag{20}$$

Since the average edge dimension over all stages is ε/n aud the average vertex dimension over the last n depths is v/n, 100SC lower bounds on V aud E can be obtained from Jensen's inequality.

Theorem 6 *The total complexity measures V, E are lower bounded in terms of the total span length complexity measures ν, c by*

$$V \ge 1 + nq^{\nu/n}$$

$$E \ge nq^{\varepsilon/n}.$$

'1'here arc also tighter lower bounds on V and E in terms of ν aud E.

Theorem 7 *Given a total span length v ore, let $\Delta\varepsilon = \varepsilon - e^{-}(n+1-e^{-})$ and $\Delta\nu = \nu - s^{-}(2n-s^{-})$, wh c re $e^{-} \le (n+1)/2$ and $s^{-} \le n/2$ are the largest integers such that $\Delta\varepsilon \ge 0$ and $\Delta\nu \ge 0$. Then*

$$V \ge \left[n + \frac{q+1}{q-1} - 2s^{-} \right] q^{s^{-}} - \frac{2}{q-1} + (q-1)q^{s^{-}}\Delta\nu$$

$$E \ge \left[n + \frac{2q}{q-1} - 2e^{-} \right] q^{e^{-}} - \frac{2q}{q-1} + (q-1)q^{e^{-}}\Delta\varepsilon.$$

This theorem follows from the observation that, for given ν or ε, a vertex or edge dimension profile such as the one in Figure 3 minimizes V or E. Notice the similarity of these lower bounds in terms of s and c with the corresponding upper bounds (1 9), (20) In terms of $s_{\max}$ and $c_{\max}$.

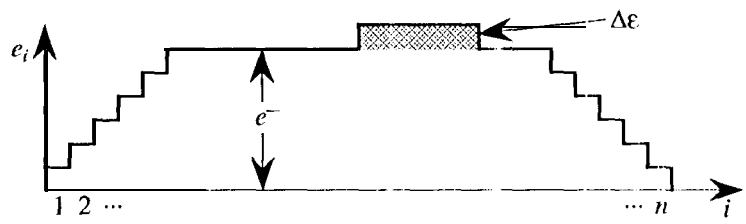


Figure 3: An edge dimension profile that minimizes E subject to a constraint, on total edge span ε.

4.2 Complexity Lower Bounds Based 011 MSGM Span Length

Every row of a generator matrix for an (n, $k, d, d^\perp$) code must have edge-sl)an length $\varepsilon_i \geq$ d and vertex-s])a]l length $\nu_i \geq d-1$. Applying this simple bound to both the code and the dual code and using the fact that $\nu^\perp =$ v = $\varepsilon - k$ leads to the following lower bounds on the span length complexity measures ν and ε.

Theorem 8 *The total lengths ν and ε of the vertex-spans and edge-spans for any $(n, k, d, d^\perp)$ code are lower bounded by*

$$\nu \geq \max\left\{k(d-1), (n-k)(d^\perp - 1)\right\}$$

$$\varepsilon \geq k + \max\left\{k(d-1), (n-k)(d^\perp - 1)\right\}.$$

Applying the Singleton bound to the inequalities in this theorem gives the weaker bounds $\nu \geq (d-1)(d^\perp - 1)$ and $\varepsilon \geq k + (d-1)(d^\perp - 1)$.

We say that a code meeting the bounds in Theorem 8 with equality is a *minimal span code.* An example is the $(n, 1, n, 2)$ repetition code. To construct a nondegenerate $(n, k, d, 2)$ binary minimal span code for any $d > 2$ and $n \geq d + (k-1)\left\lceil \frac{d}{2} \right\rceil$, let the first row of the MSGM be

$$\underbrace{111\cdots1}_{d}\underbrace{000\cdots0}_{n-d}$$

and form each successive row by cyclically shifting the previous row at least $\left\lceil \frac{d}{2} \right\rceil$ positions but not more than d positions to the right, such that the total of all the shifts is $n -$ d.

The dual of a minimal span code is also a minimal span code. These codes are not usually good in terms of distance, though they have very low complexity trellises.

The span length bounds in Theorem 8, combined with the bounds (1 7) and (18) lead to lower bounds on the complexity measures $s_{\max}, c_{\max}$ for auy (n, $k, d, d^{\perp}$) code:

$$s_{\max}(n - s_{\max}) \geq \max \{k(d - 1), (n - k)(d^{\perp} - 1)\}$$

$$c_{\max}(n + 1 - c_{\max}) \geq k + \max \{k(d - 1), (n - k)(d^{\perp} - 1)\} .$$

A slightly weaker version of this bound on $s_{\max}$ has been proved for both linear and nonlinear codes [22]. This bound implies, for instance, that the average edge dimension $c_{\max}$ can never be lower than the asymptotic coding gain led/n. We can also obtain bounds on V and E for any (72, k, d, $d^{\perp}$) code by substituting, the right hand sides of the bounds in Theorem 8 for v aud ε in Theorems 6 and 7.

4.3 Dimension/Length Profiles

We can see from the complexity measures (11) - (16) that a permutation of $\mathcal{C}$ that makes f_i and p_i large (small) wherever possible will produce a low (high) complexity trellis. It is useful, therefore, to find bounds on these quantities.

The *support* of a vector x is the set of nonzero positions in x. The support of a set of vectors is the union of the individual supports.

Definition 1 *For a given code $\mathcal{C}$ and any $0 \leq i \leq n$, let $K_i(\mathcal{C})$ be the maximum* dimension *of a linear subcode of $\mathcal{C}$ having support whose size is no greater than i. The set $\{K_i(\mathcal{C}), i = 0, \ldots, n\}$* is *called the* dimension/length profile (DLP) *[9, 10, 14, 30].*

The DLP contains tbc same information about a code as the minimum support weights [11, 17, 18], which have more recently been called tbc generalized Hamming weights (GHW) or weight hierarchy [32]. The j^{th} minimum support weight or GHW is the smallest support size of any j-dimensional linear subcode of $\mathcal{C}$.

Since the past and future subcodes $\mathcal{P}_i$ aud $\mathcal{F}_i$ arc subcodes of $\mathcal{C}$ with support size no larger than i and $n - i$, respectively, the past aud future subcode dimensions arc bounded by the DLP:

$$p_i \leq \max_{\pi \in \mathcal{S}_n} p_i(\mathcal{C}\pi) = K_i(\mathcal{C}) \tag{21}$$

$$f_i \leq K_{n-i}(\mathcal{C}). \tag{22}$$

These bounds, which also appeared in [14, cq. (1 .4)], arc tight in the following sense: for any i, there exists a permuted version of $\mathcal{C}$ that meets the bound (21), aud one that meets (22), though it may not be possible to meet both simultaneously. The DLP of a code can be used to lower bound the trellis complexity for any permutation of that code, as wc shall see in Section 4.5.

Since each $K_i(\mathcal{C})$ is associated with a linear subcode of $\mathcal{C}$, we cau usc bounds on the best possible linear codes (i.e., codes with the largest possible minimum distance) to upper bound the DLP:

Theorem 9 *For an* $(n, k, d, d^\perp)$ *code* $\mathcal{C}$ *and any* $0 \le i \le n$,

$$p_i \le K_i(\mathcal{C}) \le \overline{K}_i(n, k, d, d^\perp)$$

$$f_i \le K_{n-i}(\mathcal{C}) \le \overline{K}_{n-i}(n, k, d, d^\perp)$$

Uhfc re

$$\overline{K}_i(n, k, d, d^\perp) \stackrel{\Delta}{=} \min[k_{\max}(i, d), k - n + i + k_{\max}(n - i, d^\perp)]$$

and $k_{\max}(m, d)$ *is the largest possible dimension for any q-ary linear block code of length m and minimum distance d. The set* $\{\overline{K}_i(n, k, d, d^\perp), i = 0, ., n\}$ *is called the* upper dimension/length profile *(UDLP) for the code parameters* $(n, k, d, d^\perp)$.

Proof: Since $K_i(\mathcal{C})$ is the dimension of a code with distance d and support size not exceeding i, we have $K_i(\mathcal{C}) \le k_{\max}(i, d)$. Using Theorem 1,

$$K_i(\mathcal{C}) = \max_{\pi \in \mathcal{S}_n} p_i(\mathcal{C}\pi) = k - n + i + ym:j\sim(C17r)$$

and $f_i(\mathcal{C}^\perp \pi)$ is the dimension of a subcode of distance $d^\perp$ and support size not exceeding n — i, so $K_i(\mathcal{C}) \le k - n + i + k_{\max}(n - i, d^*)$. The theorem is a combination of these two inequalities. ■

The usc of parameters for the Lest linear codes (or bounds on such codes) to bound 1)1,1', GHW, or other quantities related to trellis complexity also appears in [7, 10, 12, 14, 19, 20, 28, 30].

Bounds based on the UDLP may be 100 SC, as it may not be possible for a single $(n, k, d, d^\perp)$ code and its dual to both have a series of subcodes, all with the maximum code dimensions. However, these bounds are important practically, because much data about the best possible codes has been tabulated [4], and in many cases, the UDLP bounds can be achieved with equality.

Since for any (n, k) code $\mathcal{C}$, p_i and f_i both reach maximum values of k ($f_0 = k$ and $p_n = k$) and can fall from these values at a maximum rate of one u nit pcr trellis stage, p_i and f_i arc lower bounded as follows:

$$K_i(\mathcal{C}) \ge p_i \ge \underline{K}_i(n, k) \stackrel{\Delta}{=} \max(0, k - n + i) \tag{23}$$

$$K_{n-i}(\mathcal{C}) \ge f_i \ge \underline{K}_{n-i}(n, k) = \max(0, k - i). \tag{24}$$

The set $\{\underline{K}_i(\mathcal{C}), i = 0, 1, \ldots n\}$ is called the *lower dimension/length profile (LDLP)* for the code parameters (n, k). The 1,1)1,1' stays at O until the last possible depth before it can rise linearly at the rate of one dimension per depth to reach its final value of k at depth n. The LDLP can be used to upper bound the complexity of a minimal trellis for an arbitrary (n, k) code.

4.4 Properties of Dimension/Length Profiles

The DLPs possess many of the same properties as the past and future subcode dimensions which they bound. For example, the monotonicity and unit increment properties (2) and (4) of $\{p_i\}$ also hold

for $K_i(\mathcal{C})$, $\overline{K}_i(\mathcal{C})$, and $\underline{K}_i(\mathcal{C})$: the increments $\overline{K}_{i+1}(n,k,d,d^\perp) - \overline{K}_i(n,k,d,d^\perp)$, $K_{i+1}(\mathcal{C}) - K_i(\mathcal{C})$, and $\underline{K}_{i+1}(n,k) - \underline{K}_i(n,k)$ must equal O or 1 for all i. Similarly, duality properties can be easily extended.

There is a convenient relationship between the DLP of a code and that of its dual, stated in [14, eq. (1 .12)] and [10, Theorem 3], which is equivalent to the duality relationship for generalized Hamming weights [32, Theorem 3]. Similar relationships hold for the upper and lower dimension/length profiles. The following lemma gives these relationships along with a proof of the DLP result, that is somewhat simpler than those that have appeared in the literature.

Lemma 4 *For all* $0 \le i \le n$, *the DLP, UDLP and LDLP satisfy the following duality relationships:*

$$K_i(\mathcal{C}^\perp) = r' - k + K_{n-i}(\mathcal{C})$$

$$\overline{K}_i(n, n-k, d^\perp, d) = i - k + \overline{K}_{n-i}(n,k,d,d^\perp)$$

$$\underline{K}_i(n, n-k) = i - k + \underline{K}_{n-i}(n,k)$$

Proof:

$$K_i(\mathcal{C}^\perp) = \max_{\pi \in \mathcal{S}_n} p_i(\mathcal{C}^\perp \pi) = \max_{\pi \in \mathcal{S}_n} (i - k + f_i(\mathcal{C}\pi)) = i - k + \max_{\pi \in \mathcal{S}_n} f_i(\mathcal{C}\pi) = r' - k + K_{n-i}(\mathcal{C})$$

where the second equality follows from Theorem 1. The other equations follow directly from the UDLP and LDLP definitions. ■

The DLP of an $(n,k,d,d^\perp)$ code $\mathcal{C}$ is related to the LDLP and UDLP as follows:

$$\overline{K}_i(n,k,d,d^\perp) \ge K_i(\mathcal{C}) \ge \underline{K}_i(n,k) \text{ for all } i \tag{25}$$

$$\overline{K}_i(n,k,d,d^\perp) = K_i(\mathcal{C}) = \underline{K}_i(n,k) \text{ if } 0 \le i \le d-1 \text{ or } n - d^\perp + 1 \le i \le n \tag{26}$$

$$\overline{K}_i(n,k,d,d^\perp) = K_i(\mathcal{C}) = \underline{K}_i(n,k) + 1 \text{ if } d \le i \le \min\left\{n-k, d + \left\lceil \frac{d}{q} \right\rceil - 1\right\} \text{ or } \max\left\{n-k, n-d^\perp - \left\lceil \frac{d^\perp}{q} \right\rceil + 1\right\} \le i \le n - d^\perp. \tag{27}$$

The DLP and UDLP are lower bounded everywhere by the 1,1)1,1' and equal the 1.1)1,1' at both ends of the interval $[0, n]$. Their range of departure from the LDLP is [d, n - $d^\perp$]. Inside this range the DLP and UDLP stay equal to each other for an additional $\lceil d/q \rceil$ depths from the left and $\lceil d^\perp/q \rceil$ depths from the right,.

The above properties follow from the definitions of the profiles, the properties of p_i stated in Section 2.2, the duality relationships in Lemma 4, and the fact that $k_{max}(n,d)$ equals the corresponding Griesmer bound when $k_{\max}(n,d) = 2$. A more thorough discussion of the DLP is contained in [10].

Example 3 Suppose C is the (6,3,3)3) shortened Hamming code whose generator matrix is given in (l). The LDLP for this code is {0,0,0,0, 1,2,3}. Since c1 = *71* — *d'-* = 3, the DLP and the UDL I' equal the LDLP except at $i = 3$. Thus the DLP and UDLP are both {O, 0,0, 1, 1,2, 3}. □

Example 4 The 1,1)1,1' of the (15,5,7,4) BCH code is {0,0,0,0,0,0,0,0,0,0,0, 1,2,3,4,5}. Applying (26) we find that the DLP and UDLP equal {0,0,0,0,0,0,0, - ,--, –, --, - ,2,3,4,5}, and (27) fills in the five missing values: the DLP and UDLP equal {0,0,0,0,0,0,0,0,1,1,1, 1,2,2,3,4,5}. ❑

In the two preceding examples, the DLP is completely determined by the code parameters n, d, and $d^{\perp}$. In fact the DLP is completely specified by (26) and (27) whenever $n - k < d + \lceil \frac{d}{q} \rceil$ and $k < d^{\perp} + \lceil \frac{d^{\perp}}{q} \rceil$. Computation of the DLP for large codes is usually much more difficult than these examples might suggest. For many codes the complete DLP is unknown, and much research has been devoted to determining partial DLP (or equivalently, GHW) information for codes [5, 7, 12, 18, 29, 33].

Example 5 Suppose $\mathcal{C}$ is a (48, 24, 12, 12) self- dual code (e.g., the quadratic residue code with these parameters). Applying (25)- (27) produces the DLP bounds for $\mathcal{C}$ shown in Figure 4. ❑

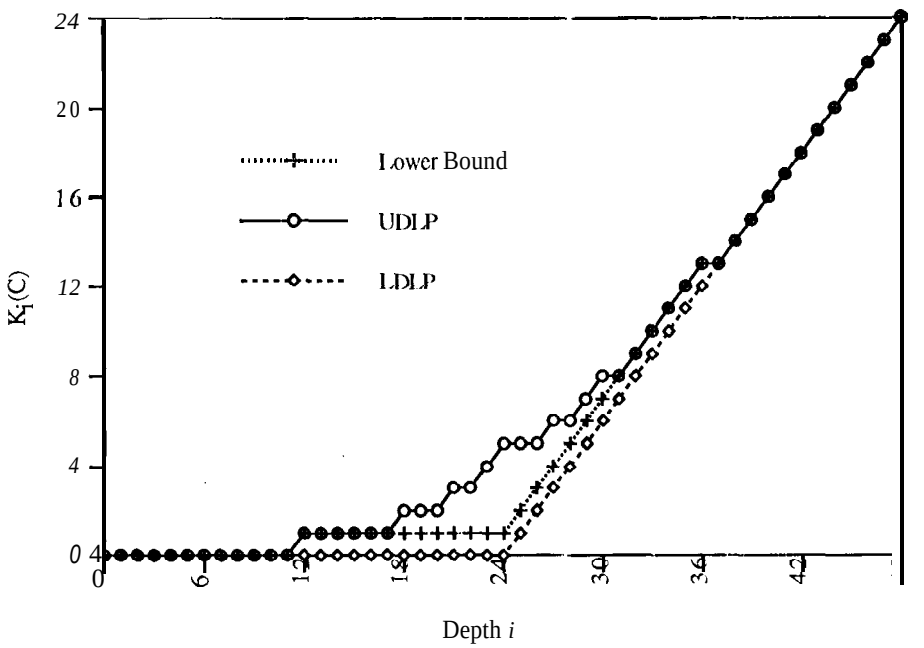


Figure 4: DLP bounds for a (48,24,12) self-dual code

4.5 Complexity Bounds from Dimension/1 ength Profiles

The DLP bounds (21) and (22) and complexity definitions (11) - (16) lead to simple bounds on trellis complexity that are useful when the DLP of a given code is known. These bounds can be tightened slightly by using the additional fact that the vertex and edge dimensions must be nonnegative everywhere. (In fact, if a trellis converges to a single vertex at depth i, i.e., $v_i = 0$ for some r' # O or n, then $\mathcal{C}$ is a direct sum code, see Appendix A.)

Theorem 10 *The complexity measures for the minimal trellis $T(\mathcal{C}\pi)$ corw.spending to any permutation π of a given (n, k) code $\mathcal{C}$ arc lower bounded by:*

$$s_{\max}(\mathcal{C}\pi) \geq \max_{i\in[0,n]} (k - K_i(\mathcal{C}) - K_{n-i}(\mathcal{C})) \tag{28}$$

$$c_{\max}(\mathcal{C}\pi) \geq \max_{i\in[1,n]} (k - K_{i-1}(\mathcal{C}) \; K_{n-i}(\mathcal{C})) \tag{29}$$

$$\varepsilon(\mathcal{C}\pi) \geq \sum_{i=0}^{n} \max\{0, k - K_i(\mathcal{C}) - K_{n-i}(\mathcal{C})\} \tag{30}$$

$$V(\mathcal{C}\pi) \geq \sum_{i=0}^{\mathbf{n}} q^{\max\{0 \;\; \text{k-K,(C)-} \;\; K_{n-i}(\mathcal{C})\}} \tag{31}$$

$$E(\mathcal{C}\pi) \geq \sum_{i=1}^{n} q^{\max\{0, k - K_{i-1}(\mathcal{C}) \cdot K_{n-i}(\mathcal{C})\}} \tag{32}$$

$$M(\mathcal{C}\pi) \geq \frac{1}{q}\sum_{i=1}^{n}[K_i(\mathcal{C}) - K_{i-1}(\mathcal{C})]q^{\max\{0,k-K_{i-1}(\mathcal{C})-} \;\; - \;\; \text{`(c)}\}. \tag{33}$$

The DLP bound (28) on state complexity has been derived in [1 O, 30, 23]. Some of the bounds in Theorem 10 can be improved slightly when $\mathcal{C}$ is nondegener ate because this condition implies that $c_i \geq 1$.

The UDLP bound (Theorem 9) leads to similar lower bounds on trellis complexity that apply to all c.odes with given code parameters.

Theorem 11 *The complexity measures for the minimal trellis 7(C) representing any (n, $k, d, d^{\perp}$) code $\mathcal{C}$ are 10 wer bounded by:*

$$s_{\max}(\mathcal{C}) \geq \max_{i\in[0,n]} \left[k - \overline{K}_i(n,k,d,d^{\perp}) - \overline{K}_{n-i}(n,k,d,d^{\perp})\right] \tag{34}$$

$$c_{\max}(\mathcal{C}) \geq \max_{i\in[1,n]} \left[k - \overline{K}_{i-1}(n,k,d,d^{\perp}) - \overline{K}_{n-i}(n,k,d,d^{\perp})\right] \tag{35}$$

$$\mathrm{E(C)} \geq \sum_{i=1}^{n} \max\{\mathrm{O}, \; \mathrm{k} - \overline{K}_{i-1}(n,k,d,d^{\perp}) - \overline{K}_{n-i}(n,k,d,d^{\perp})\} \tag{36}$$

$$V(\mathcal{C}) \geq \sum_{i=0}^{n} q^{\max\left\{0,k-\overline{K}_i(n,k,d,d^{\perp})-\overline{K}_{n-i}(n,k,d,d^{\perp})\right\}} \tag{37}$$

$$E(\mathcal{C}) \geq \sum_{i=1}^{n} \mathrm{g}^{\max\left\{0,k-\overline{K}_{i-1}(n,k,d,d^{\perp})-\overline{K}_{n-i}(n,k,d,d^{\perp})\right\}} \tag{38}$$

$$M(\mathcal{C}) \geq \frac{1}{q}\sum_{i=1}^{n}\left[\overline{K}_i(n,k,d,d^{\perp}) - \overline{K}_{i-1}(n,k,d,d^{\perp})\right] q^{\max\left\{0,k-\overline{K}_{i-1}(n,k,d,d^{\perp})-\overline{K}_{n-i}(n,k,d,d^{\perp})\right\}} \tag{39}$$

Finally, the LDLP bounds (23) (24) lead immediately to simple explicit upper bounds on the various complexity measures that apply to all codes with given length and dimension.

Theorem 12 *The complexity measures for the minimal trellis* 7 (C) *corresponding to any* (n,k) *code* $\mathcal{C}$ *are upper bounded by:*

$$s_{max}(\mathcal{C}) \leq \min(k, n - k) \tag{40}$$

$$e_{max}(\mathcal{C}) \leq \min(k, n - k + 1) \tag{41}$$

$$\varepsilon(\mathcal{C}) \leq k(n - k + 1) \tag{42}$$

$$V(\mathcal{C}) \leq n + \frac{q+1}{q-1} - 2\min(k, n-k)\, q^{\min(k,n-k)} \frac{2}{q-1} \tag{43}$$

$$E(\mathcal{C}) \leq n + \frac{2q}{q-1} - 2\min(k, n-k+1)\, q^{\min(k,n-k+1)} \frac{2q}{q-1} \tag{44}$$

$$M(\mathcal{C}) \leq \frac{1}{q-1} + \max(0, 2k-n)\, q^{\min(k,n-k)} \frac{1}{q-1}. \tag{45}$$

The inequality (40) is the well-known Wolf bound [34]. Note that (17) (20) are tighter than (42)- (44), except when (40) and (41) are met with equality, in which case the bounds are the same.

The derivation of the inequalities in Theorems 10, 11, and 12 is rather straightforward, with the exception of the bounds on M (equations (33), (39), and (45)). These bounds can be derived using the same arguments as used in the proof of Theorem 13 in the next section.

5 Best and Worst Trellises

5.1 Uniform Comparability

In general, to determine which of two minimal trellises is less complex, we must first choose the relevant complexity measure. however, in some cases one trellis may be simpler than another at every stage and depth with respect to all of the complexity measures simultaneously.

Definition 2 *For two* (n,k) *codes* $\mathcal{C}_1$, $\mathcal{C}_2$ *having minimal trellises* $\mathcal{T}(\mathcal{C}_1)$ *and* $\mathcal{T}(\mathcal{C}_2)$, *we say that* $\mathcal{T}(\mathcal{C}_1) \preceq \mathcal{T}(\mathcal{C}_2)$ *if* $p_i(\mathcal{C}_1) \geq p_i(\mathcal{C}_2)$ *and* $f_i(\mathcal{C}_1) \geq f_i(\mathcal{C}_2)$ *for all* i. *If either* $\mathcal{T}(\mathcal{C}_1) \preceq \mathcal{T}(\mathcal{C}_2)$ *or* $\mathcal{T}(\mathcal{C}_2) \preceq \mathcal{T}(\mathcal{C}_1)$, *then the two trellises are* uniformly comparable.

The binary relation $\preceq$ defines a partial ordering on any set of codes with the same length and dimension. If $\mathcal{T}(\mathcal{C}_1) \preceq \mathcal{T}(\mathcal{C}_2)$ and $\mathcal{T}(\mathcal{C}_2) \preceq \mathcal{T}(\mathcal{C}_1)$ then the two minimal trellises have equivalent complexity, though they may not have the same structure. For example, for the code $\mathcal{C}_2$ of example 2, $\mathcal{T}(\mathcal{C}_2) \preceq \mathcal{T}(\mathcal{C}_2^{\perp})$ and $\mathcal{T}(\mathcal{C}_2^{\perp}) \preceq \mathcal{T}(\mathcal{C}_2)$ but $\mathcal{C}_2 \neq \mathcal{C}_2^{\perp}$.

Note that if $\mathcal{T}(\mathcal{C}_1) \preceq \mathcal{T}(\mathcal{C}_2)$ then at every depth and stage $\mathcal{T}(\mathcal{C}_1)$ has no more vertices or edges than $\mathcal{T}(\mathcal{C}_2)$, but the converse is not necessarily true. For example, the codes with generator matrices $\begin{bmatrix} 1100 \\ 0011 \end{bmatrix}$ and $\begin{bmatrix} 1100 \\ 0111 \end{bmatrix}$ are not uniformly comparable, even though the first has a minimal trellis that is at least as simple at every stage than that of the second. We define comparability in terms

of past and future dimensions rather than edge and vertex dimensions because this gives a closer connection to the dimension/length profiles.

Theorem 13 *If* $\mathcal{T}(\mathcal{C}_1) \preceq \mathcal{T}(\mathcal{C}_2)$, *then all of the following trellis complexity measures for* $\mathcal{C}_1$ *are upper bounded by those for* $\mathcal{C}_2$:

1. *maximum state complexity:* $s_{\max}(c,) \leq s_{\max}(\mathcal{C}_2)$
2. *total span lengths:* $\varepsilon(\mathcal{C}_1) \leq \varepsilon(\mathcal{C}_2)$, $\nu(\mathcal{C}_1) \leq \nu(\mathcal{C}_2)$
3. *total vertices:* $V(\mathcal{C}_1) \leq V(\mathcal{C}_2)$
4. *total edges:* $E(\mathcal{C}_1) \leq E(\mathcal{C}_2)$
5. *total number of path mergers:* $M(\mathcal{C}_1) \leq M(\mathcal{C}_2)$

Proof: Inequalities 1 - 4 follow immediately from the definitions. It remains to show that $M(\mathcal{C}_1) \leq M(\mathcal{C}_2)$. From (16),

$$M = \frac{1}{q}\sum_{i=1}^{n} r_i q^{e_i} = \frac{1}{q}\sum_{i=1}^{n} r_i q^{k-f_i-p_{i-1}} = q^{k-1}\sum_{i=1}^{n} r_i q^{-f_{iq}} \text{- P-- I}$$

Now $r_i = 1$ in precisely the k places where p_i is incremented, so the nonzero values of $r_i q^{-p_{i-1}}$ are $q^0, q^{-1}, \ldots, q^{-(k-1)}$, which gives

$$M = q^{k-1}\sum_{j=0}^{k-1} q^{-j} q^{-f_{R_j}}$$

where R_j is the position of the j^{th} 1 in $(r_1, r_2, \ldots \text{m})$. Uniform comparability implies $p_i(\mathcal{C}_1) \geq p_i(\mathcal{C}_2)$, thus $R_j\ (C_,) \leq R_j\ (C])$, and

$$f_{R_j(\mathcal{C}_1)}(\mathcal{C}_1) \geq f_{R_j(\mathcal{C}_1)}(\mathcal{C}_2) \geq f_{R_j(\mathcal{C}_2)}(\mathcal{C}_2)$$

so

$$M(\mathcal{C}_1) = q^{k-1}\sum_{j=0}^{k-1} q^{-j} q^{-f_{R_j(\mathcal{C}_1)}(\mathcal{C}_1)} \leq q^{k-1}\sum_{j=0}^{k-1} q^{-j} q^{-f_{R_j(\mathcal{C}_2)}(\mathcal{C}_2)} = M(\mathcal{C}_2).$$

■

If two minimal trellises are not uniformly comparable then the choice of the less complex t rellis may depend on which of the complexity measures is used as the criterion.

Uniform comparability is a very strong property that is not guaranteed to exist between auy two trellises. Our motivation for defining it and studying its consequences lies in the correspondingly strong results obtained for the problem of finding a minimal trellis in the first place, i.e., finding the least complex trellis that represents a fixed permutat ion of a fixed code. As shown in [27], the minimal trellis is uniformly less complex at every stage and depth than any other trellis that represents the code.

We define four categories of best and worst minimal trellises based on uniform comparability:

Definition 3 *For a fixed code $\mathcal{C}$, a permutation π^* and the corresponding minimal trellis $\mathcal{T}(\mathcal{C}\pi^*)$ are*

- uniformly efficient *if* $\mathcal{T}(\mathcal{C}\pi^*) \preceq \mathcal{T}(\mathcal{C}\pi)$ *for all* $\pi \in \mathcal{S}_n$
- uniformly inefficient *if* $\mathcal{T}(\mathcal{C}\pi) \preceq \mathcal{T}(\mathcal{C}\pi^*)$ *for all* $\pi \in \mathcal{S}_n$.

Definition 4 *An (n, $k, d, d^\perp$) code $\mathcal{C}^*$ and its corresponding minimal trellis 7 (C') is*

- uniformly concise *if* $\mathcal{T}(\mathcal{C}^*) \preceq$ T(C) *for all (n, k, d, d*) codes $\mathcal{C}$*
- uniformly full *if T(C)* $\preceq \mathcal{T}(\mathcal{C}^*)$ *for all (n, k) codes $\mathcal{C}$*

If a minimal trellis is uniformly efficient or uniformly concise, we can drop the qualifier "minimal" and refer to it simply as a uniformly efficient trellis or a uniformly concise trellis, respectively. As shown later in Theorem 22, the two worst-case categories, uniformly inefficient and uniformly full, turn out to be equivalent.

The **inclusion of $d^\perp$ in the above definition elucidates** symmetries that are hidden by consideration of only n, k, and d. First, it preserves duality relationships, as we shall see below in Theorem 14. Second, from a practical point of view, d and $d^\perp$ have symmetric impact on the potential trellis complexity. There also appears to be a deep connection between d and $d^\perp$ for good codes: often when d *is* large, $d^\perp$ must also be large, e.g., the extended Hamming codes and MDS codes.

A direct consequence of Theorem 1 is that Uniform comparability of codes and their duals are equivalent:

Theorem 14 *$\mathcal{T}(\mathcal{C}_1) \preceq \mathcal{T}(\mathcal{C}_2)$ if and only if $\mathcal{T}(\mathcal{C}_1^\perp) \preceq \mathcal{T}(\mathcal{C}_2^\perp)$. Consequently:*

1. *A permutation π^* is uniformly efficient for $\mathcal{C}$ if and only if π^* is uniformly efficient for $\mathcal{C}^\perp$*
2. *A permutation π^* is uniformly inefficient for $\mathcal{C}$ if and only if π^* is uniformly inefficient for $\mathcal{C}^\perp$ [14, Theorem 1].*
3. *$\mathcal{C}^*$ is uniformly concise if and only if $\mathcal{C}^{*\perp}$ is uniformly concise*
4. *$\mathcal{C}^*$ is uniformly full if and only if $\mathcal{C}^{*\perp}$ is uniformly full.*

In the next sections we show that the trellis complexity bounds derived in Section 4.5 are met exactly for the four categories of extremal minimal trellises.

5.2 Best Permutations

The following theorem shows that uniformly efficient trellises are those that achieve the DLP bounds in (21), (22), and Theorem 10 with equality.

Theorem 15 *A permutation π^* is uniformly efficient for a nondegenerate code $\mathcal{C}$ if and only if $\mathcal{C}\pi^*$ meets the DLP bounds (21) and (22) with equality, i.e.,*

$$p_i(\mathcal{C}\pi^*) = K_i(\mathcal{C}) \text{ and } f_i(\mathcal{C}\pi^*) = K_{n-i}(\mathcal{C}) \text{ for all } i.$$

This guarantees that $\mathcal{C}\pi^*$ meets all of the lower bounds on complexity (28) - (33) with equality. Conversely, if $\mathcal{C}\pi^*$ meets any one of the lower bounds (30) - (32) with equality, then π^* is a uniformly efficient permutation for $\mathcal{C}$.

The proof of this theorem is given in Appendix B.

Theorem 15 shows that uniformly efficient permutations, which are defined in terms of trellis comparability, turn out to be the same as "efficient" [10] or "strictly optimum" [14] orderings which were defined in terms of the DLP bounds. Note that a code may not have a permutation that meets these conditions.

A uniformly efficient permutation, if it exists, is not unique: If π^* is uniformly efficient for $\mathcal{C}$, then so is the reverse of π^*, and in fact the number of uniformly efficient permutations must be at least as large as the automorphism group of the code. There may also be different permutations that are uniformly efficient and produce distinct MSGMs for the code.

Example 6 The following MSGMS produce uniformly efficient minimal trellises for the (8,4,4,4) extended Hamming code:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & x & \overline{x} & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & \overline{y} & y & 1 & 0 \end{bmatrix}_H, \qquad \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & x & \overline{x} & \overline{y} & y & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

where in each case x and y can be assigned arbitrarily, and $\overline{x}, \overline{y}$ denote the complement of x and y respectively. ❑

Even though uniform efficiency is a very strong property to require of a trellis, there are many codes that have uniformly efficient permutations. For example, the standard permutation of any Reed-Muller code is uniformly efficient [14, Theorem 2]. Additional examples of uniformly efficient codes are given in Section 5.3, which lists trellises that are both uniformly efficient and uniformly concise.

We now include some theoretical results that impose necessary conditions on uniformly efficient permutations.

Theorem 16 *Suppose $\mathcal{C}$ is a code that has some uniformly efficient permutation π^*. Then for any i, j such that $i + j \leq n$,*

$$K_{i+j}(\mathcal{C}) \geq K_i(\mathcal{C}) + K_j(\mathcal{C}).$$

Proof: $\mathcal{P}_i(\mathcal{C}\pi^*)$ and $\mathcal{F}_i(\mathcal{C}\pi^*)$ are disjoint subcodes of $\mathcal{C}\pi^*$. Since $n-j \geq i, \mathcal{F}_i(\mathcal{C}\pi^*) \supseteq \mathcal{F}_{n-j}(\mathcal{C}\pi^*)$, SO $\mathcal{P}_i(\mathcal{C}\pi^*)$ and $\mathcal{F}_{n-j}$ (Cm") are disjoint subcodes of $\mathcal{C}\pi^*$. Since π^* is uniformly efficient for $\mathcal{C}, K_i(\mathcal{C}) = p_i(\mathcal{C}\pi^*) = \dim(\mathcal{P}_i(\mathcal{C}\pi^*))$ and $K_j(\mathcal{C}) = f_{n-j}(\mathcal{C}\pi^*) = \dim(\mathcal{F}_{n-j}(\mathcal{C}\pi^*))$. Consequently $\mathcal{P}_i(\mathcal{C}\pi^*) \cup \mathcal{F}_{n-j}(\mathcal{C}\pi^*)$ is a linear subcode of $\mathcal{C}\pi^*$ of dimension $K_i(\mathcal{C}) + K_j(\mathcal{C})$ and support not exceeding $i + j$. ■

Theorem 17 *If π^* is a uniformly efficient permutation for an (71, k, d, $d^\perp$) code $\mathcal{C}$, then $\mathcal{C}\pi^*$ contains codewords of the form $X^d 0^{n-d}$, $0^{n-d} X^d$ and $\mathcal{C}^\perp \pi^*$ contains codewords of the form $X^{d^\perp} 0^{n-d^\perp}$, $0^{n-d^\perp} X^{d^\perp}$ where 0^j denotes j consecutive zeros, and X^j denotes some sequence of j non. zero symbols from $GF(q)$.*

Proof: Uniform efficiency implies $p_i(\mathcal{C}\pi^*) = K_i(\mathcal{C})$ for all i, so $p_d(\mathcal{C}\pi^*) = K_d(\mathcal{C}) = 1$, i.e., $\mathcal{C}\pi^*$ has $q-1$ codewords of weight d and support confined to the first d positions, thus $X^d 0^{n-d} \in$ Cm". Similarly, $f_{n-d}(\mathcal{C}\pi^*) = K_d(\mathcal{C}) = 1$ establishes that $0^{n-d} X^d \in \mathcal{C}\pi^*$. The rest follows because, from Theorem 14, π^* is uniformly efficient for $\mathcal{C}$ if and only if it is uniformly efficient for $\mathcal{C}^\perp$. ■

Corollary 18 *If $\mathcal{C}$ is a binary $(n, k, d, d^\perp)$ code that has some uniformly efficient permutation π^*, then $\min(d, d^\perp)$ must be even.*

Proof: From Theorem 17, $1^d 0^{n-d} \in \mathcal{C}\pi^*$ and $1^{d^\perp} 0^{n-d^\perp} \in \mathcal{C}^\perp \pi^*$, but if $\min(d, d^\perp)$ is odd then these sequences cannot be orthogonal. ■

Here are some examples of specific codes which cannot, have a uniformly efficient permutation, according to the preceding necessary conditions:

Example 7 Let $\mathcal{B}$ be **the (3,1,3,2)** repetition code, and let $\mathcal{C}$ be the direct sum code $\mathcal{C} = \mathcal{B} \oplus \mathcal{B}^\perp$. Then $\mathcal{C}$ has DLP **{0,0, 1,2,2,2,3}** so $K_6(\mathcal{C}) < K_3(\mathcal{C}) + K_3(\mathcal{C})$, hence by Theorem 16, $\mathcal{C}$ has no uniformly efficient permutation. As we shall see later, this code also has no uniformly inefficient permutation. □

Example 8 By Corollary 18, the (23, 12,7,8) Golay code has no uniformly efficient permutation, neither dots the $(2^m - 1, 2^m - m - 1, 3)$ Hamming code for any $m \geq 3$. Consequently, no nontrivial perfect binary linear code has a uniformly efficient permutation. □

Example 9 The (1 0,5,4,4) formally self-dual code with generator matrix

$$\begin{bmatrix} 1000000111 \\ 0100010011 \\ 0010011001 \\ 0001011100 \\ 0000101110 \end{bmatrix}$$

has weight enumerator $1 + 15x^4 + 15x^6 + x^{10}$. If this code has a uniformly efficient permutation then by Theorem 17 it must have two disjoint codewords of weight four, hence it must also have a codeword of weight 8. But the weight enumerator reveals that no such codeword exists. We shall see in Section 5.3, however, that there is a (10,5,4,4) code that has a uniformly efficient permutation. ❑

The same argument shows that the (15,5,7,4) BCH code has no uniformly efficient permutation.

Example 10 The (15,7,5,4) BCH code does not have a uniformly efficient permutation, because this code does not have a minimum weight codeword whose support contains the support of a minimum weight dual codeword. (This code is small enough that this fact can be verified by exhaustive search.) ❑

The preceding examples show that many codes lack uniformly efficient permutations. However, for many such codes there exists some permutation that simultaneously minimizes all of the trellis complexity measures. For example, if $\mathcal{C}_1$ and $\mathcal{C}_2$ are two different codes with uniformly efficient permutations, it **is not usually true** that **the direct** sum code $\mathcal{C}_1 \oplus \mathcal{C}_2$ has a uniformly efficient permutation (e.g., code $\mathcal{C}$ of example 7), even though the corresponding trellis $\mathcal{T}(\mathcal{C}_1 \oplus \mathcal{C}_2)$ cannot be improved upon according to any of the complexity measures. Thus, it can be argued that $\mathcal{T}(\mathcal{C}_1 \oplus \mathcal{C}_2)$ is efficient, though not uniformly so. Another example is the (7,4) Hamming code, which is sufficiently small that we can verify by exhaustive search that there are permutations that are optimal with respect to all of the complexity measures despite not being uniformly efficient.

For self-dual codes, Theorem 3 tells us that there is always a single permutation that simultaneously minimizes E, V, and M. We suspect that not every code has a permutation that simultaneously minimizes all of the complexity measures, though we do not yet know of an example that confirms this conjecture.

5.3 Best Codes

Uniformly concise codes are optimum in a rather strong sense. Not only do they have an efficient permutation, but they also minimize all of the trellis complexity measures compared to all codes with the same parameters. The following theorem shows that codes that achieve the bounds in Theorems 9 and 11 with equality are uniformly concise.

Theorem *19 An $(n, k, d, d^\perp)$ code $\mathcal{C}^*$ is uniformly concise if the dimensions of its past and future subcodes meet the bounds in Theorem 9 with equality, i.e.,*

$$p_i(\mathcal{C}^*) = \overline{K}_i(n, k, d, d^\perp) \text{ and } f_i(\mathcal{C}^*) = \overline{K}_{n-i}(n, k, d, d^\perp) \text{ for all } i.$$

in this case $\mathcal{C}^*$ meets all of the lower bounds on complexity (34) - (39) with equality. Conversely, if $\mathcal{T}(\mathcal{C}^*)$ meets any of the bounds (36) through (38) with equality, then $\mathcal{C}^*$ is uniformly concise.

We suspect that a code can be uniformly concise without meeting the bounds of Theorem 9 with equality.

Table 2 lists known uniformly concise binary codes. In each case, the complexity values listed are the lowest possible for any code with the same parameters. From Theorem 14, the dual of each code is also uniformly concise. Generator matrices for many of these codes are given in Appendix C. All of the rate $\frac{1}{2}$ codes in the table are either self-dual, or have duals that are permuted versions of the original code.

The first entry in the table comes from applying the DLP properties to some of the minimal span codes discussed in Section 4.2. These codes all have dimension $k \leq 3$ and include the $(n, 1, n)$ repetition codes. Since minimal span codes minimize ε, every minimal span code is either uniformly concise, or no uniformly concise code exists with the same parameters. We shall see in Example 11 that not all minimal span codes are uniformly concise.

The second entry in Table 2 follows from Theorem 20 below. The remaining examples are obtained by discovering a column permutation that meets the 1)1,1'. Uniformly efficient permutations for the (24,12,8,8) extended Golay code and the (32,16,8,8) second-order Reed-Muller code have been reported elsewhere [8, 14], and it is easy to verify by reference to the table in [4] that the trellises for these permutations are uniformly concise. The other examples have not been reported elsewhere and are briefly discussed here or in Appendix C. The last two entries are somewhat trivial, since they are direct-sums of other uniformly concise codes. However, applying the direct-sum construction) to uniformly concise codes does not always produce another- uniformly concise code.

The (48,24,12,12) code with generator matrix given in Appendix C is uniformly concise. This code is an optimally permuted version of the (48,24, 12, 12) quadratic residue code.

Uniform conciseness for first-order Reed-Muller codes and extended Hamming codes is established in the following theorem.

Theorem 20 *All $(2^m, m+1, 2^{m-1}, 4)$ first-order Reed-Muller codes and their duals, the $(2^m, 2^m - m - 1, 4, 2^{m-1})$ extended Hamming codes, are uniformly concise.*

Proof: This can be proved using the explicit GHW for first order Reed-Muller codes derived by Wei [32, Theorem 5] together with the result of Kasami et. al. [14, Theorem 2] that the standard permutation is uniformly efficient. The result for extended Hamming codes then follows from duality (Theorem 13). ■

There are also examples of code parameters $(n, k, d, d^\perp)$ for which **no** uniformly concise trellis can exist.

Example 11 The (9,4,3,2) minimal span code has DLP {0,0,0,1,1,2,2,3,3,4}. If $\mathcal{C}$ is the direct sum of the (6,3,3,3) shortened Hamming code, and the (3,1,3,2) repetition code, then $\mathcal{C}$ is a (9,4,3,2) code with superior 1)1,1' {0,0,0,1,1,2,3,3,3,4} but larger edge span length ε. This proves that there is no (9,4,3,2) uniformly concise code. ❑

code parameters	E	V	M	$s_{\max}$	$e_{\max}$	ε	Comment
$(d+(k-1)\lceil\frac{d}{2}\rceil, k, d, 2)$	$2kd$	$2+2k(d-1)$	$2k-1$	$\min\{2, d-2\}$	2	kd	minimal span codes, $d>2$, $k\leq 3$
[dual]	$4k(d-2)+4$	$2+2k(d-1)$	$2k(d-3)+3$	2	2	$k(d-2)+n$	
$(2^m, 1+m, 2^{m-1}, 4)$	(a)	(b)	$3(2^{m-1})-1$	m	m	$2+(m-1)2^m$	first-order Reed-Muller $\mathcal{R}(1,m)$
[dual]	(c)	(d)	(e)	m	$m+1$	$m(2^m-2)$	extended Hamming
$(24, 12, 8, 8)$ self-dual	3580	2686	895	9	9	136	extended Golay $\mathcal{G}_{24}$
$(32, 16, 8, 8)$ self-dual	6396	4798	1599	9	9	202	second-order Reed-Muller
$(48, 24, 12, 12)$ self-dual	860156	645118	115039	16	16	502	quadratic residue
$(10, 5, 4, 4)$ formally self-dual	60	46	15	3	3	24	see Appendix C
$(12, 6, 4, 4)$ formally self-dual	76	58	19	3	3	30	see Appendix C
$(16, 4, 8, 2)$	88	78	11	3	3	36	see Appendix C
[dual]	132	78	55	3	4	44	
$(20, 6, 8, 4)$	236	206	31	4	4	66	see Appendix C
[dual]	348	206	143	4	5	74	
$(24, 7, 8, 4)$	300	262	39	4	4	82	see Appendix C
[dual]	444	262	183	4	5	92	
$(24, 8, 8, 4)$	364	302	63	5	5	86	see Appendix C
[dual]	476	302	175	5	5	94	
$(40, 7, 16, 4)$	940	878	63	5	5	170	see Appendix C
[dual]	1628	878	751	5	6	196	
$(16, 8, 4, 4)$ self-dual	88	67	22	3	3	36	$\mathcal{R}(1,3) \oplus \mathcal{R}(1,3)$
$(48, 24, 8, 8)$ self-dual	7160	5371	1790	9	9	272	$\mathcal{G}_{24} \oplus \mathcal{G}_{24}$

Table 2: Some known uniformly concise binary codes. Codes are grouped with their duals, which are also uniformly concise. Expressions too big to fit into the table: (a) $\frac{(2^{2m+1}+2^4)}{3}-4$, (b) $\frac{(2^{2m+1}+2^4)}{3}-3(2^{m-1})-2$, (c) $\frac{(2^{2m+2}+2^5)}{3}-4-3(2^{m+1})$, (d) $\frac{(2^{2m+1}+2^4)}{3}-3(2^{m-1})-2$, (e) $\frac{(2^{2m+1}+2^4)}{3}-9(2^{m-1})-1$. Complexity expressions for first order Reed-Muller and extended Hamming codes are valid for $m \geq 3$, except $e_{\max}=3$ when $m=3$.

From definitions 3 and ,4, clearly if $\mathcal{C}\pi^*$ is uniformly concise then π^* must be uniformly efficient for $\mathcal{C}$.

Example 12 The U])],]' of any (6,3,2,2) code is {O, O, 1,2,2,2, 3}, but we saw in example 7 that this profile doesn't satisfy the constraints of Theorem 16 for uniformly efficient permutations. So no such code has a uniformly efficient permutation, hence no (6,3,2,2) code can be uniformly concise. ❑

Example 13 'J'here is no binary (18,9,6,6) uniformly concise code, (e.g., the quadratic residue code with these parameters). If there were such a code, Theorem 17 tells us that x 816012 must be in the optimally permuted code and its dual. The UDLP of auy (18,9,6,6) code begins with {O, 0,0,0,0,0,1 ,1,1,2,.. .}, which is the beginning of the DLP for the quadratic residue code. This implies $r_9 = 1$ for such a code, i.e., there must be some codeword y whose span ends in position 9. Now d = 6 implies $|x+y| \geq 6$ aud $|y| \geq 6$, so y must be of the form $X^6 1^3 0^9$ where X^6 denotes some permutation of $1^3 0^3$. But in this case, x and y arc not orthogonal, contradicting the fact that $x \in \mathcal{C}^\perp$. D

This argument also shows that there is no binary $(n, k, 2m, 2m)$ code that meets the U])),]' bounds when $k > 1$ and m is odd, e.g., the (42, 21, 10, 10) quadratic residue code. The $\mathcal{R}(r,m)$ Reed-Muller codes when (m = 6, r = 2, 3), (m = 7, r = 2,3,4) are also codes that do not meet the UDLP bounds. This is established by comparing the UDLP bounds to the known optimal permutations for the Reed-Muller codes.

Results such as the examples above and Theorems 16 and 17 illustrate that iu many instances the U 1)1,1' bounds on complexity arc not tight. Au area of further research is to produce tighter bounds on trellis complexity based on the code parameters (n, $k, d, d^\perp$).

5.4 Worst Minimal Trellises

The following theorems show that uniformly inefficient and uniformly full minimal trellises are tbc same as the trellises that achieve the 1,1)1,1' bounds with equality.

Theorem 21 *An (n, k) code $\mathcal{C}$ is uniformly full if and only if the dimensions of the past and future subcodes of $\mathcal{C}$ meet the bounds (23), (24) with equality, i.e.,*

$$p_i(\mathcal{C}) = \max(0, k - n + i) \text{ and } f_i(\mathcal{C}) = \max(0, k - i) \text{ for all } i.$$

in this case $\mathcal{C}$ meets all of the upper bounds on complexity (40) - (45) with equality. Conversely, if $\mathcal{C}$ meets any one of the upper bounds (42) - (44) with equality, then $\mathcal{C}$ is uniformly full.

Theorem 22 *A minimal trellis $T(\mathcal{C}\pi^*)$ is uniformly full if and only if π^* is a uniformly inefficient permutation of $\mathcal{C}$.*

Proof: If $\mathcal{C}$ is k-dimensional, any generator matrix for $\mathcal{C}$ has k linearly independent columns. Let π_1 and π_2 be two permutations which place k independent columns in the first and last k positions, respectively. Then $p_i(\mathcal{C}\pi_2) = \max(0, k - n + i)$ and $f_i(\mathcal{C}\pi_1) = \max(0, k - i)$, i.e., π_2 achieves the LDLP bound on the past subcode dimensions, and π_1 achieves it for the future subcode dimensions. If π^* is uniformly inefficient, then $p_i(\mathcal{C}\pi^*) \leq p_i(\mathcal{C}\pi_2)$ and $f_i(\mathcal{C}\pi^*) \leq f_i(\mathcal{C}\pi_1)$, i.e., π^* must achieve the LDLP bounds on both the past and the future. Thus, a trellis is uniformly inefficient if and only if both the first k columns and last k columns of the corresponding generator matrices are linearly independent. This proves that a uniformly inefficient minimal trellis is also uniformly full. The reverse implication is trivial. ■

Many codes have uniformly inefficient trellises in their standard permutations. For example, the minimal trellises for all cyclic, extended cyclic, and shortened cyclic codes are uniformly inefficient [15, 20]. However, not every code has a uniformly inefficient permutation, e.g., code C of example 7.

Additional examples of uniformly inefficient trellises are given in the following two theorems.

Theorem 23 *A self-dual code always has a uniformly inefficient permutation*

Proof: By Theorem 2, any $(2k, k)$ self-dual code has k stages of simple expansions and k stages of simple mergers. The k columns of the generator matrix corresponding to the expansion stages form a linearly independent set, as do the k columns corresponding to the merger stages. Therefore, any permutation which groups all k of the expansion columns followed by all k of the merger columns is uniformly inefficient. ■

Theorem 24 *If and only if a code is maximum distance separable (MDS), every permutation π is uniformly inefficient and the corresponding trellis complexity measures equal the upper bounds in (40) - (45).*

This theorem follows from the fact that a code is maximum distance separable if and only if every subset of k columns of its generator matrix is linearly independent. A peculiar consequence of Theorem 24 is that every permutation of an MDS code is also uniformly efficient, as noted by Forney [10]. This observation emphasizes that uniform efficiency is only a relative measure of trellis complexity.

6 Conclusion

In this paper we have attacked the trellis complexity problem by first considering the minimal span generator matrix for a fixed permutation of a code. McEliece [27] showed that the so-called minimal trellis indeed minimizes not only the maximum state dimension of the trellis but also a whole gamut of complexity measures. Here we have augmented the list of reasonable complexity measures and

interrelated them. We have also illustrated the connection between the complexity measures and the four primitive structures of a minimal trellis for a nondegenerate code.

The trellis complexity analysis for a fixed code generalizes naturally to similar results for codes allowed to vary over a domain of optimization. We identified two useful domains, the set of permutations of a given code and the set of all codes with given code parameters. Within each domain we defined uniformly best and worst minimal trellises that are guaranteed to simultaneously minimize or maximize all of the complexity measures. We showed that it is easy to generalize the bounds on maximum state complexity derived by other authors from the dimension/length profile of a code to similar bounds on all the complexity measures over each optimization domain. Furthermore, if a minimal trellis attains the bounds for some of the complexity measures, it must necessarily be uniformly extremal, but this is not true for the simpler measures of maximum state or edge dimension considered by other authors. This lends further credence to the argument that a measure of total complexity (such as the total number of edges) is more useful than a measure of maximum complexity [27].

Unlike the case of a fixed permutation of a given code, uniformly best and worst minimal trellises are not guaranteed to exist within the larger domains of optimization. However, we demonstrated the usefulness of the concepts by presenting several examples of uniformly best trellises, most notably the optimum permutation of the (48,24) quadratic residue code, heretofore unknown. Conversely, by deriving some necessary existence conditions, we also identified some cases for which uniformly extremal minimal trellises cannot exist.

We developed a series of useful relationships between the trellis complexity of a code and that of its dual, again in a natural progression first from a fixed code and then to larger code domains. This approach yields many of the same results obtained by other authors for dimension/length profiles or generalized Hamming weights, but it emphasizes that all the duality results stem from fundamental minimal trellis relationships valid for a fixed permutation of a code. In fact, we have argued that the symmetry of the constraints imposed by the code and its dual on trellis complexity is so fundamental that the minimum distance of the dual code should be included as one of the intrinsic code parameters that limits achievable complexity. The duality relationships lead to interesting connections among several of the complexity measures for the special case of self-dual codes.

Appendices

A Direct Sum Codes

Definition 5 (Direct Sum Codes) *[24, p. 76] If $\mathcal{C}_1$ and $\mathcal{C}_2$ are $(n_1, k_1, d_1, d_1^\perp)$ and $(n_2, k_2, d_2, d_2^\perp)$ linear block codes respectively, then the* direct sum *code $\mathcal{C}$ (denoted $\mathcal{C} = \mathcal{C}_1 \oplus \mathcal{C}_2$) is the set of*

all codewords of the form $c_1|c_2$ (i. c., c_1 followed by c_2) where $c_1 \in \mathcal{C}_1$ and $C_2 \in \mathcal{C}_2$. $\mathcal{C}$ is an $(n_1 + n_2, k_1 + k_2, \min(d_1, d_2), \min(d_1^{\perp}, d_2^{\perp}))$ linear block code. We refer to $\mathcal{C}_1$ and $\mathcal{C}_2$ as component codes *of $\mathcal{C}$. Direct sum codes of more than two codes are defined in the obvious manner.*

For direct sum codes, the DLP can be computed from the DLPs of its component codes in the following manner.

Lemma 5 *If $\mathcal{C} = \mathcal{C}_1 \oplus \mathcal{C}_2$ then*

$$K_i(\mathcal{C}) = \max_{m \in [0,i]} K_m(\mathcal{C}_1) + K_{i-m}(\mathcal{C}_2)$$

where we interpret $K_j(\mathcal{B})$ to equal $\dim(\mathcal{B})$ when j exceeds the length of code $\mathcal{B}$.

Proof: Let $\mathcal{Q}$ denote the linear subcode of $\mathcal{C}$ associated with $K_i(\mathcal{C})$. Since $\mathcal{C}$ is a direct sum code, $\mathcal{Q} = \mathcal{Q}_1 \oplus \mathcal{Q}_2$, for some subcodes $\mathcal{Q}_1 \subseteq \mathcal{C}_1, \mathcal{Q}_2 \subseteq \mathcal{C}_2$. Then $K_i(\mathcal{C}) = \dim(\mathcal{Q}) = \dim(\mathcal{Q}_1) + \dim(\mathcal{Q}_2)$ and $\mathrm{supp}(\mathcal{Q}) = \mathrm{supp}(\mathcal{Q}_1) + \mathrm{supp}(\mathcal{Q}_2) \leq i$. Letting $m = \mathrm{supp}(\mathcal{Q}_1)$, clearly we maximize $\dim(\mathcal{Q})$ if $\dim(\mathcal{Q}_1) = K_m(\mathcal{C}_1), \mathrm{supp}(\mathcal{Q}_2) \leq i - m$, and $\dim(\mathcal{Q}_2) = K_{i-m}(\mathcal{C}_2)$. ■

The definition of uniform efficiency has the desirable property that, if $\mathcal{T}(\mathcal{C})$ is uniformly efficient, then the minimal trellis for the direct sum of $\mathcal{C}$ with itself, $\mathcal{T}(\mathcal{C} \oplus \mathcal{C})$, is also uniformly efficient.

B Proof of Theorem 15

Proof: For each $0 \leq i \leq n$, we can always find permutations π_p, π_f such that $p_i(\mathcal{C}\pi_p) = K_i(\mathcal{C})$ and $f_i(\mathcal{C}\pi_f) = K_{n-i}(\mathcal{C})$. By definition, if π^* is uniformly efficient then $\mathcal{T}(\mathcal{C}\pi^*) \preceq \mathcal{T}(\mathcal{C}\pi_p)$ and $\mathcal{T}(\mathcal{C}\pi^*) \preceq \mathcal{T}(\mathcal{C}\pi_f)$, so $p_i(\mathcal{C}\pi^*) \geq K_i(\mathcal{C})$ and $f_i(\mathcal{C}\pi^*) \geq K_{n-i}(\mathcal{C})$. This combined with (21) and (22) proves the first statement.

If $\mathcal{C}\pi^*$ is uniformly efficient then by the above argument clearly $\mathcal{C}\pi^*$ meets (28) - (33) with equality, so it remains only to show the converse for (30) - (32). If $\mathcal{C}\pi^*$ meets (32) then for each $1 \leq i \leq n$, either (a) $c_i = 0$ or (b) $p_{i-1} = K_{i-1}(\mathcal{C})$ and $f_i = f_{n-i+1}(\mathcal{C})$. If (a) occurs then $\mathcal{C}$ is nondegenerate, so (b) must **hold** everywhere, hence π^* is uniformly efficient for $\mathcal{C}$.

If $\mathcal{C}\pi^*$ meets (30) or (31) then for each $0 < i < n$ either (a) $v_i = 0$ or (b) $p_i = K_i(\mathcal{C})$ and $f_i = K_{n-i}(\mathcal{C})$. We have just established that if (b) holds for all $0 < i < n$ then π^* is uniformly efficient for $\mathcal{C}$. We will now show that whenever (a) holds, (b) must also hold. If (a) occurs at one or more depths, then $\mathcal{C}\pi^*$ is a direct sum code and every depth i where $v_i = 0$ marks a boundary between component codes. Let the first of the component codes be denoted by $\mathcal{B}$, an $(n_{\mathcal{B}}, k_{\mathcal{B}}, d_{\mathcal{B}}, d_{\mathcal{B}}^{\perp})$ code. For all $i < n_{\mathcal{B}}$, $p_i(\mathcal{B}) = p_i(\mathcal{C}\pi^*) = K_i(\mathcal{C}) > K_i(\mathcal{B})$. But from (21), $p_i(\mathcal{B}) < K_i(\mathcal{B})$, so

$$K_i(\mathcal{B}) = p_i(\mathcal{B}). \tag{46}$$

Similarly, for all $i < n_{\mathcal{B}}$, $f_i(\mathcal{B}) + k - k_{\mathcal{B}} = f_i(\mathcal{C}\pi^*) = K_{n-i}(\mathcal{C}) \geq k - k_{\mathcal{B}} + K_{n_{\mathcal{B}}-i}(\mathcal{B})$ thus $f_i(\mathcal{B}) \geq K_{n_{\mathcal{B}}-i}(\mathcal{B})$. But from (22), $f_i(\mathcal{B}) \leq K_{n_{\mathcal{B}}-i}(\mathcal{B})$, therefore

$$K_{n_{\mathcal{B}}-i}(\mathcal{B}) = f_i(\mathcal{B}). \tag{47}$$

Equations (46) and (47) establish that $\mathcal{T}(\mathcal{B})$ is uniformly efficient. This procedure can be repeated for the other component codes in $\mathcal{C}\pi^*$.

Since $\mathcal{C}\pi^*$ (and by implication also $\mathcal{B}$) is assumed to be nondegenerate, $d_{\mathcal{B}} > 1$, $d_{\mathcal{B}}^{\perp} > 1$, and $n_{\mathcal{B}} > 1$ so depth $n_{\mathcal{B}} - 1$ cannot be a boundary between component codes, i.e., (b) holds at this depth, thus

$$K_{n_{\mathcal{B}}-1}(\mathcal{C}) = p_{n_{\mathcal{B}}-1}(\mathcal{C}\pi^*) = p_{n_{\mathcal{B}}-1}(\mathcal{B}) = K_{n_{\mathcal{B}}-1}(\mathcal{B}) = k_{\mathcal{B}} - 1.$$

This last equality follows from (26). Since the DLP is incremented by no more than one unit each time index,

$$K_{n_{\mathcal{B}}}(\mathcal{C}) \leq K_{n_{\mathcal{B}}-1}(\mathcal{C}) + 1 = k_{\mathcal{B}}.$$

Also $K_{n_{\mathcal{B}}}(\mathcal{C}) \geq p_{n_{\mathcal{B}}}(\mathcal{C}\pi^*) = k_{\mathcal{B}}$, therefore $K_{n_{\mathcal{B}}}(\mathcal{C}) = k_{\mathcal{B}} = p_{n_{\mathcal{B}}}(\mathcal{C}\pi^*)$. A similar argument establishes that $K_{n-n_{\mathcal{B}}}(\mathcal{C}) = k - k_{\mathcal{B}} = f_{n_{\mathcal{B}}}(\mathcal{C}\pi^*)$, so (b) holds at depth $n_{\mathcal{B}}$. Repeating this at each boundary between component codes establishes that (b) holds at all indices hence π^* is uniformly efficient for $\mathcal{C}$. ■

C Some Uniformly Concise Codes

in this appendix we give minimal span generator matrices for several uniformly concise codes in their optimal permutations. Since $\mathcal{C}$ is uniformly concise if and only if its dual is, in each case we give the generator matrix for the smaller of $\mathcal{C}$ and $\mathcal{C}^{\perp}$.

Example 9 showed a (10,5,4,4) formally self-dual code that was not even uniformly efficient. Another formally self-dual (10,5,4,4) code has MSGM

$$\begin{bmatrix} 1111000000 \\ 0011110000 \\ 0000111100 \\ 0000001111 \\ 1010101010 \end{bmatrix}$$

and is uniformly concise. This is a quasi-cyclic [24, p. 506] code with weight enumerator $1 + 10x^4 + 16x^5 + 5x^8$.

Other uniformly concise codes include a (12,6,4,4) formally self-dual code:

$$\begin{bmatrix} 111100000000 \\ 001111000000 \\ 000011110000 \\ 000000111100 \\ 000000001111 \\ 010101010101 \end{bmatrix}$$

a (16,4,8,2) code:

$$\begin{bmatrix} 1111111100000000 \\ 0000111111110000 \\ 0000000011111111 \\ 0011001111001100 \end{bmatrix}$$

(this MSGM can be obtained by deleting a row from the MSGM for the (1 6,5,8,4) first-order Reed-Muller code), a (20,6,8,4) code:

$$\begin{bmatrix} 11111111000000000000 \\ 00001111111100000000 \\ 00000000111111110000 \\ 00000000000011111111 \\ 00110011110011001100 \\ 01010101101010101010 \end{bmatrix}$$

a (24,7,8,4) code:

$$\begin{bmatrix} 111111110000000000000000 \\ 000011111111000000000000 \\ 000000001111111100000000 \\ 000000000000111111110000 \\ 0000000000000000] 1111111 \\ 001100111100110011001100 \\ 010101011010101010101010 \end{bmatrix}$$

a (24,8,8,4) code:

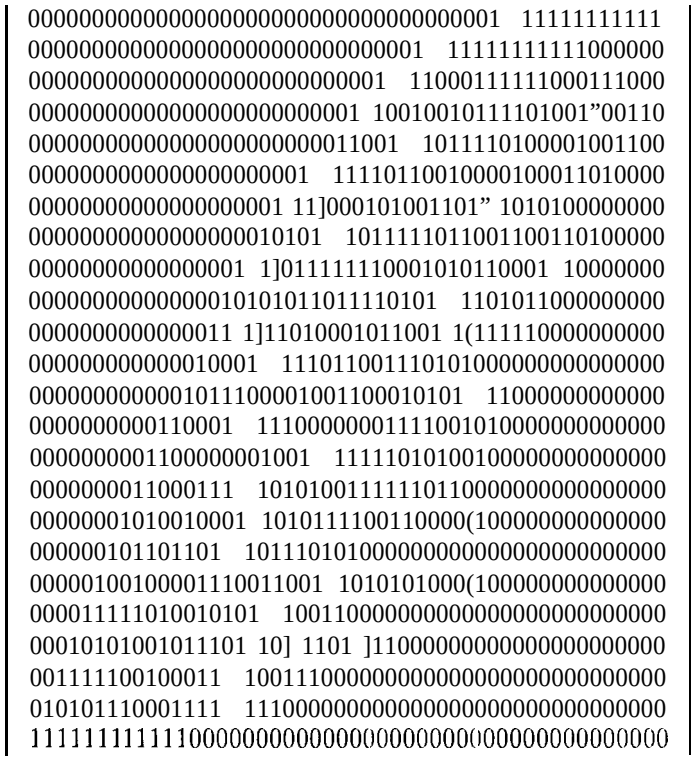
$$\begin{bmatrix} 111111110000000000000000 \\ 000011111111000000000000 \\ 000000001111111100000000 \\ 000000000000111111110000 \\ 000000000000000011111111 \\ 001100111010110011001100 \\ 010101011100101000000000 \\ 000000000011010110101010 \end{bmatrix}$$

aud a (40,7,16,4) c.ode:

$$\begin{bmatrix} 111111111111111 \; 1000000000000000000000000 \\ 0000000011111111 \; 1111111100000000000000000 \\ 0000000000000001 \; 1111111111111100000000 \\ 00000000000000000000001 \; 111111]11111111 \\ 000011110000111111110000111100001 \; 1110000 \\ 00110011001100]111001100110011001 \; 1001100 \\ 01010101010101011010101010101010101010 \end{bmatrix}$$

Finally, here is an optionally permuted generator matrix for the (48,24,1 2,12) self-dual quadratic

residue code:



References

[1] **1,.** R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inform. Theory* , vol. IT-20, no., 2, pp. 284-287, March 1974.

[2] Y. Berger and Y. Be'ery, "hounds on the Trellis Size of Linear Block Codes," *IEEE Trans. inform. Theory*, vol. 39, pp. 203-209, 1993.

[3] Y. Berger and Y. Be'ery, "On the Trellis Complexity of Composite-Length Cyclic Codes," *Proc. 1994 IEEE International Symposium on Information Theory,* Trondheim, p. 338.

[4] A. E. Brouwer and T. Verhoeff, "An updated table of Minimum-Distance Bounds for Binary Linear Codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 662-677, 1993.

[5] 11. Chung, "The Second Generalized hamming Weight of Double-Error Correcting Binary BCH Codes and their Dual Codes," *Lecture Notes in* Complier *Science*, vol. 539, pp. 1 18-129, 1991.

[6] S. Dolinar, 1,. Ekroot, A. Kiely, W. Lin, and R. J. McEliece, "The Permutation Trellis Complexity of Linear Block codes," *Proc. 32nd Annual Allerton Conferen cc on Communication, Control, and Computing*, Allerton, Illinois, October 1994.

[7] G. 1.. Feng, K. K. Tzeng, and V. K. Wei, "On the Generalized II amming Weights of Several Classes of Cyclic Codes," *IEEE Trans. Inform. Theory*, vol. 38, pp. 1125-1130, 1992.

[8] G. D. Forney, "Coset Codes- Part II: Binary Lattices and Related Codes, (Appendix A)" *IEEE Trans. Inform. Theory*, vol. 34, pp.1 152- 1 187, 1988.

[9] G. 1). Forney, "Density/Length l'refiles and Trellis ('omplexity of Linear Block Codes and Lattices," *Proc. 1994 IEEE International Symposium on Information ~'leery,* Trondheim, p. 339.

[1 0] G. D. Forney, "Dimension /Length Profiles and Trellis Complexity of Linear Block Codes," submitted to *IEEE Trans. Inform. Theory* .

[11] 'l'. Helleseth, 'l'. Kløve, and J. Mykkeltveit, "The Weight Distribution of Irreducible cyclic Codes with Block Lengths $n_1((q^l - 1)/N))$," *Discrete Math., vol. 18,* pp. *179-211, 1977.*

[12] T. Helleseth, 'J'. Kløve, and Ø. Ytrehus, "Generalized Hamming Weights of Linear Codes," *IEEE Trans. Inform. Theory*, vol. 38, pp. 1133-1140, 1992.

[13] B. Honary, G. Markarian, and M. Darnell, "Low-Complexity Trellis Decoding of Linear Block Codes," *Proc.* 1994 *IEEE International Symposium on Information Theory,* Trondheim, p. 340.

[14] 'l'. Kasami, T. Takata, T. Fujiwara, and S. Lin "On the Optimum Bit Orders with Respect to the State Complexity of Trellis Diagrams for Binary Linear Codes," *IEEE Trans. Inform. Theory*, vol. *39, pp. 242-245, 1993.*

[15] 'l'. Kasami, T. Takata, T. Fujiwara, and S. Lin "On Complexity of Trellis Structure of Linear Block Codes," *IEEE Trans. inform. Theory*, vol. *39,* pp. *1057-1064,* 1993.

[16] T. Kasami, 'l'. Takata, T. Fujiwara, and S. Lin, "On Structural Complexity of the 1,-Section Minimal Trellis Diagrams for Binary Linear Block Codes," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, vol.* E76A, no. 9, pp. 1411-1421, 1993.

[17] '1'. Kløve, "Support Weight Distribution of Linear Codes," *Discrete Mathematics,* vol. 106, pp. 311-316, Sept. 1992.

[18] T. Kløve, "Minimum Support, Weights of Binary Codes," *IEEE Trans. Inform. Theory*, vol. *39,* pp. *648-654, 1993.*

[19] A. D. Kot and C. Leung, "On the Construction and Dimensionality of Linear Block Code Trellises," *Proc. 1993 IEEE international Symposium on Information Theory,* Budapest, P. 291.

[20] F. R. Kschischang and V. Sorokine, "On the Trellis Structure of Block Codes," *Proc. 1994 IEEE International Symposium on Information Theory*, Trondheim, p. 337.

[21] F. R. Kschischang ant] V. Sorokine, "On the Trellis Structure of Block Codes," submitted to *IEEE Trans. inform. Theory*.

[22] A. Lafourcade and A. Vardy, "Asymptotically Good Codes Have Infinite Trellis Complexity," *IEEE Trans. Inform. Theory*, to appear.

[23] A. Lafourcade and A. Vardy, "Lower Bounds on Trellis Complexity of Block Codes," preprint.

[24] F. J. MacWilliams and N. J. A. Sloane, "The Theory of Error-Correcting Codes," Amsterdam: North-Holland, 1977.

[25] J. L. Massey, "(foundations and Methods of Channel Coding," in *Proc. of the Int. Conf. on info. Theory and Systems*, vol. 65, NTG-Fachberichte, Sept. 1978.

[26] R. J. McEliece, "The Viterbi Decoding Complexity of Linear Block Codes," *Proc. 1994 IEEE International Symposium on Information Theory*, Trondheim Norway, p. 341.

[27] R. J. McEliece, "On The BCJR Trellis for Linear Block Codes," submitted to *IEEE Trans. inform. Theory*.

[28] D. J. Muder, "Minimal Trellises for Block Codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1049-1053, 1988.

[29] G. van der Geer and M. van der Vlugt, "On Generalized Hamming Weights of BCH codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 543-546, 1994.

[30] A. Vardy and Y. Be'ery, "Maximum-Likelihood Soft Decision Decoding of BCH Codes," *IEEE Trans. Inform. Theory* vol. 40, pp. 546-554, 1994.

[31] H. N. Ward, "Divisible Codes," *Arch. Math.*, vol. 36, pp. 485-494, 1991.

[32] V. K. Wei, "Generalized Hamming Weights for Linear Codes," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1412-1418, 1991.

[33] V. K. Wei, "On the Generalized Hamming Weights of Product Codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1709-1713, 1993.

[34] J. K. Wolf, "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 76-80, 1978.

Figure Captions

Figure 1. A minimal trellis for the (6,3,3) shortened Hamming code.

Figure 2. Minimal trellises (a) $\mathcal{T}(\mathcal{C}_1)$ and $\mathcal{T}(\mathcal{C}_1^{\perp})$ ($\mathcal{C}_1$ is self-clual), (b) $\mathcal{T}(\mathcal{C}_2)$, (c) $\mathcal{T}(\mathcal{C}_2^{\perp})$.

Figure 3. An edge dimension profile that minimizes E subject to a constraint on total edge span ε.

Figure 4. 1)1,1' bounds for a (48,24,12) self-dual code.

'1'able 1. Dual primitive structures.

Table 2. Some known uniformly concise binary codes. Codes arc grouped with their duals, which arc also uniformly concise. Expressions too big to fit into the table: (a) $\frac{(2^{2m+1}+2^4)}{3} - 4$, (b) $\frac{(2^{2m+1}+2^4)}{3} - 3(2^{\prime\prime\prime}\text{-}]) - 2$, (C) $\frac{(2^{2m+2}+2^5)}{3}$ -4-3(2^{m+1}), (d) $\frac{(2^{2m+1}+2^4)}{3}$ -3($2^{\prime\prime\prime\prime}$) -2, (C) $\frac{(2^{2m+1}+2^4)}{3} - 9(2^{m-1}) - 1$. Complexity expressions for first order Reed-Muller and extended Hamming codes arc valid for $m \geq 3$, except $c_{max} = 3$ when $m = 3$